
neotechnology
value in relationships

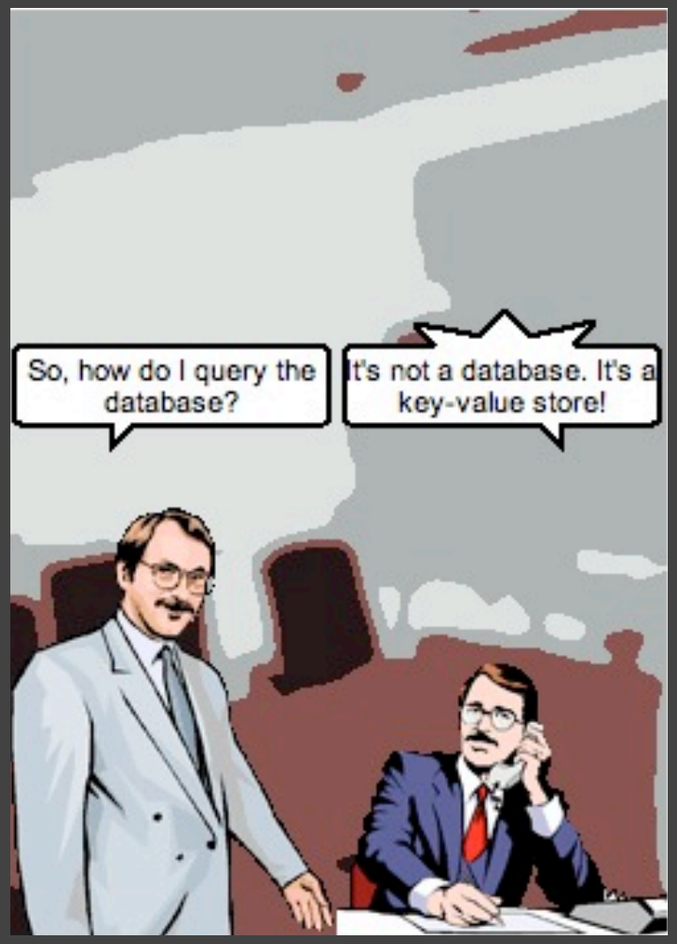


Image credit: http://browsertoolkit.com/fault-tolerance.png

Image credit: http://browsertoolkit.com/fault-tolerance.png

Image credit: http://browsertoolkit.com/fault-tolerance.png

NOSQL

- an overview - goto; con 2010

Emil Eifrem
CEO, Neo Technology

@emileifrem
emil@neotechnology.com

So what's the plan?

- Why NOSQL?
- The NOSQL landscape
- NOSQL challenges
- Conclusion

First off: the name

- WE ALL HATES IT, M'KAY?

NOSQL is NOT...

NOSQL is NOT...

- NO to SQL

NOSQL is NOT...

- NO to SQL
- NEVER SQL

NOSQL is simply

Not Only SQL

Surprise: 44% Of Business IT Pros Never Heard Of NoSQL

They should. It's fast, resilient, and often cheaper than conventional databases. Plus, it's the backbone of many Web 2.0 sites.

By Charles Babcock
InformationWeek

September 18, 2010 12:01 AM (From the September 20, 2010 Issue)

NOSQL - Why now?

Four trends

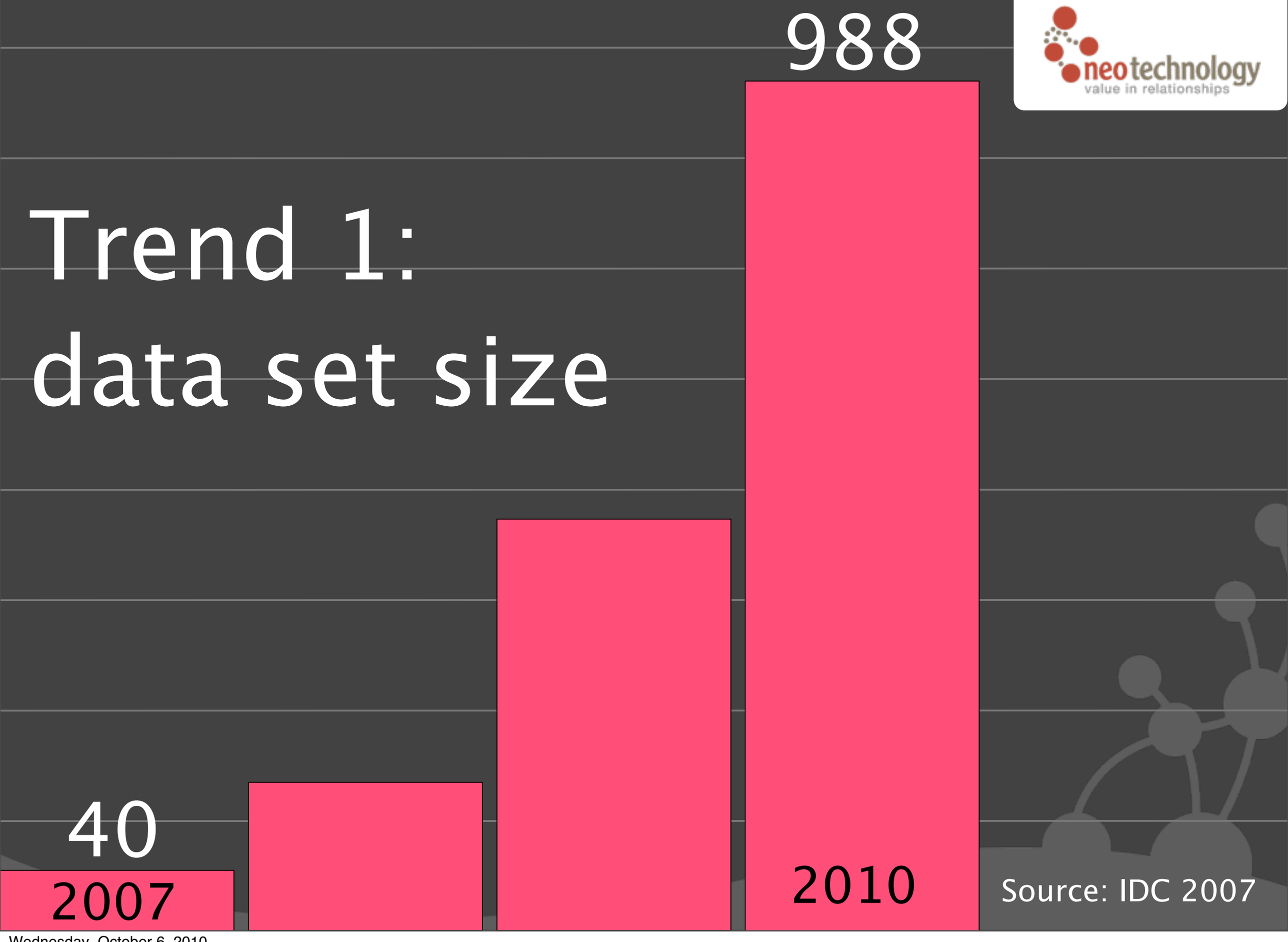
Trend 1:
data set size

40

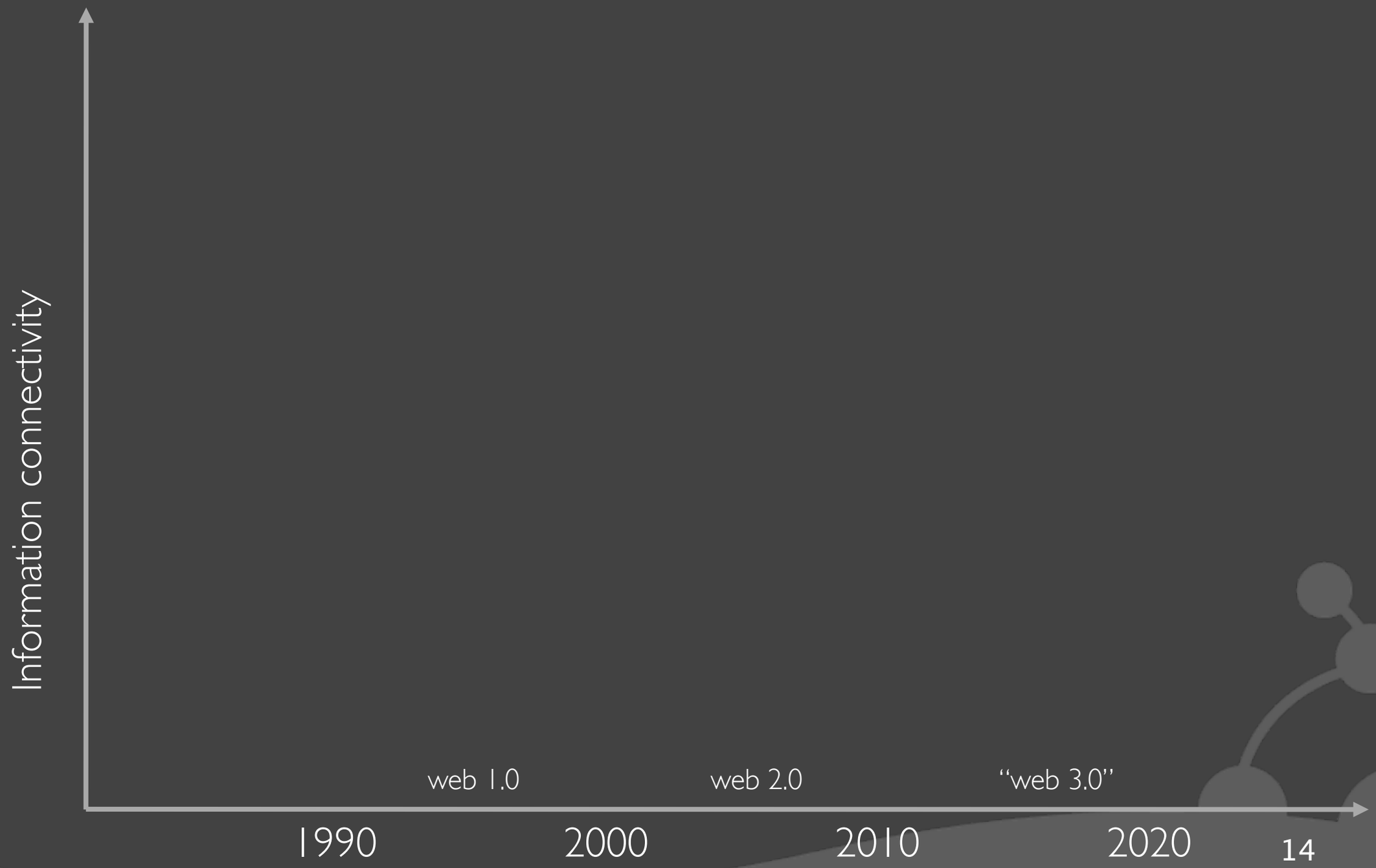
2007

Source: IDC 2007

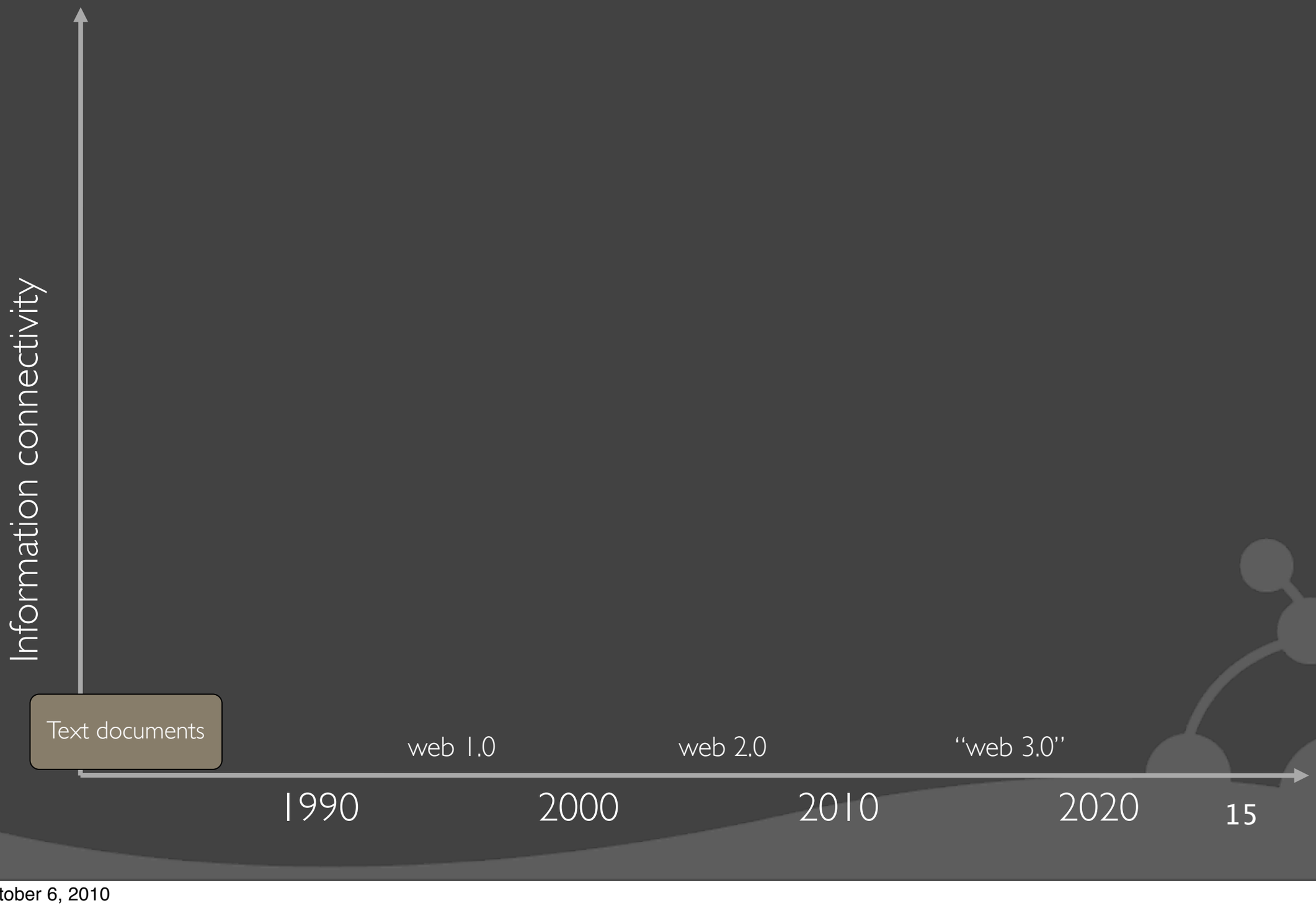
Trend 1: data set size



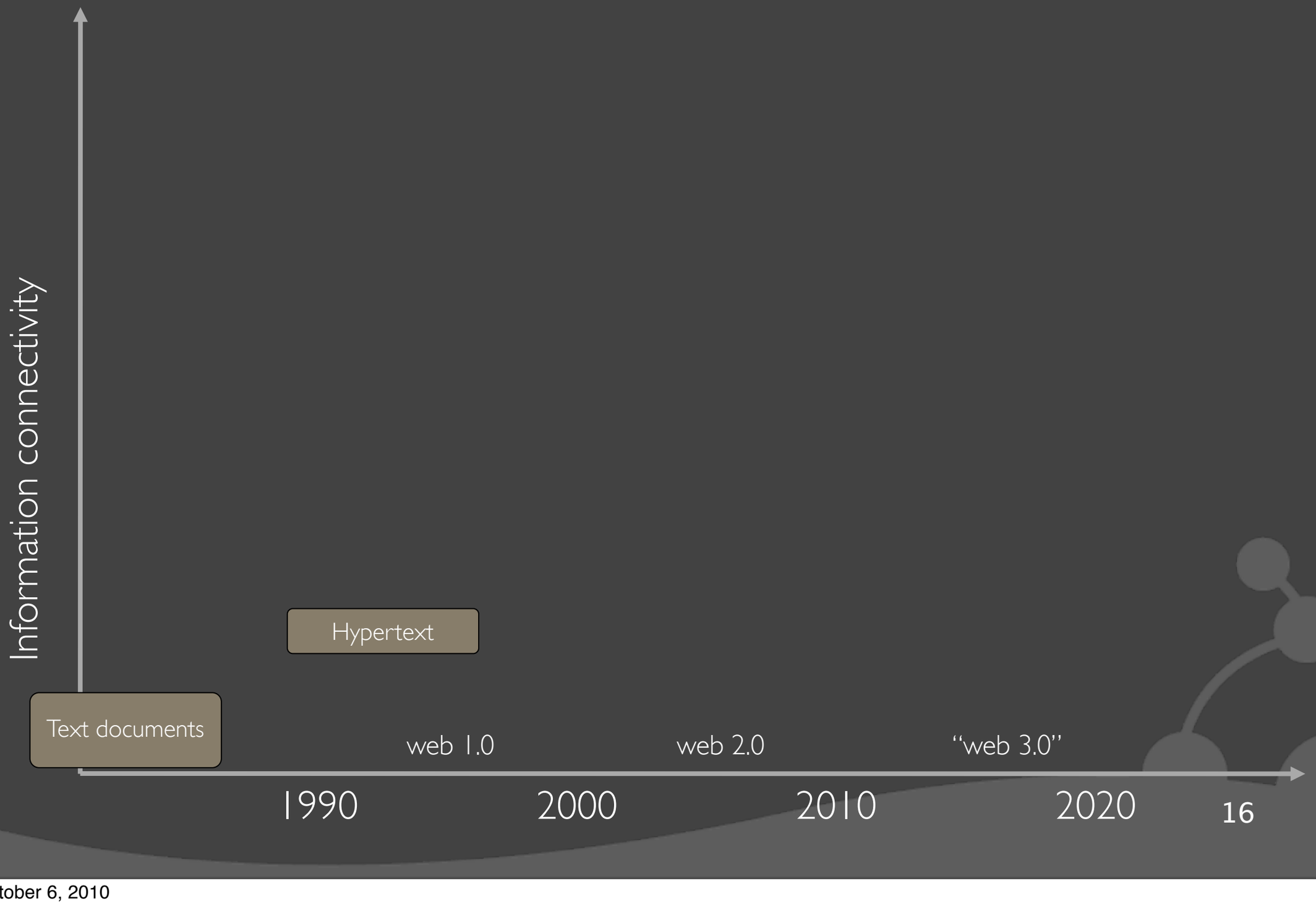
Trend 2: Connectedness



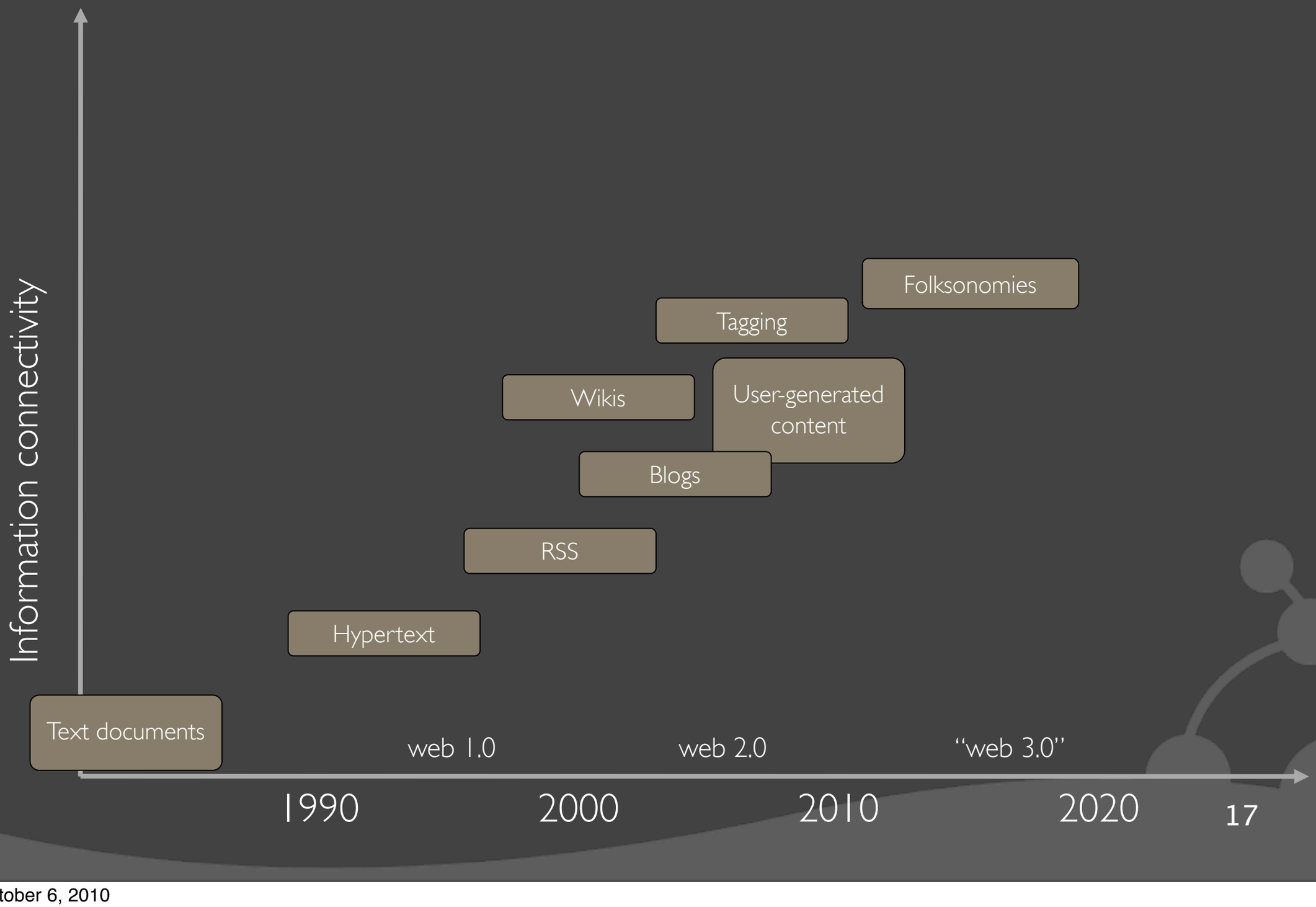
Trend 2: Connectedness



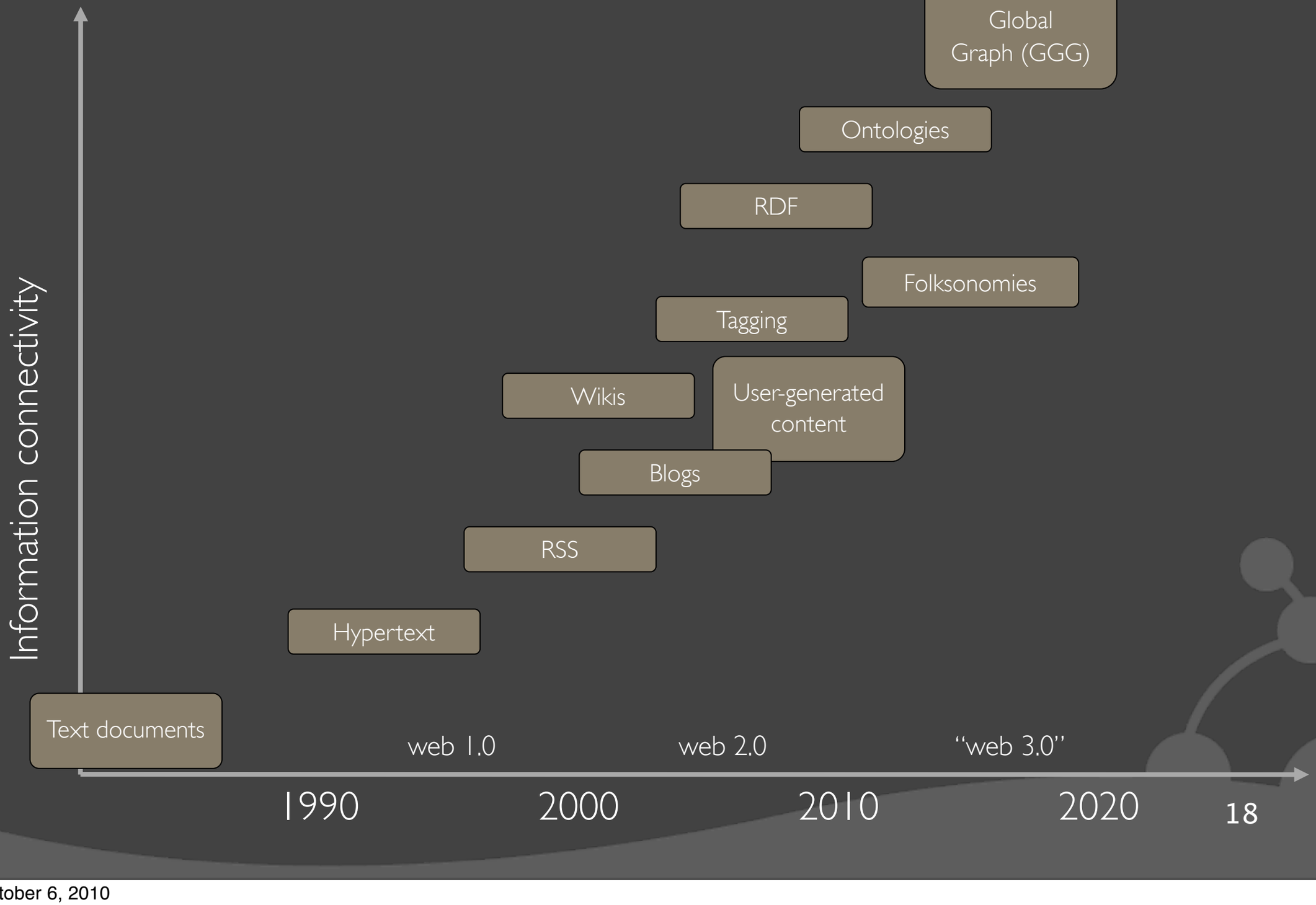
Trend 2: Connectedness



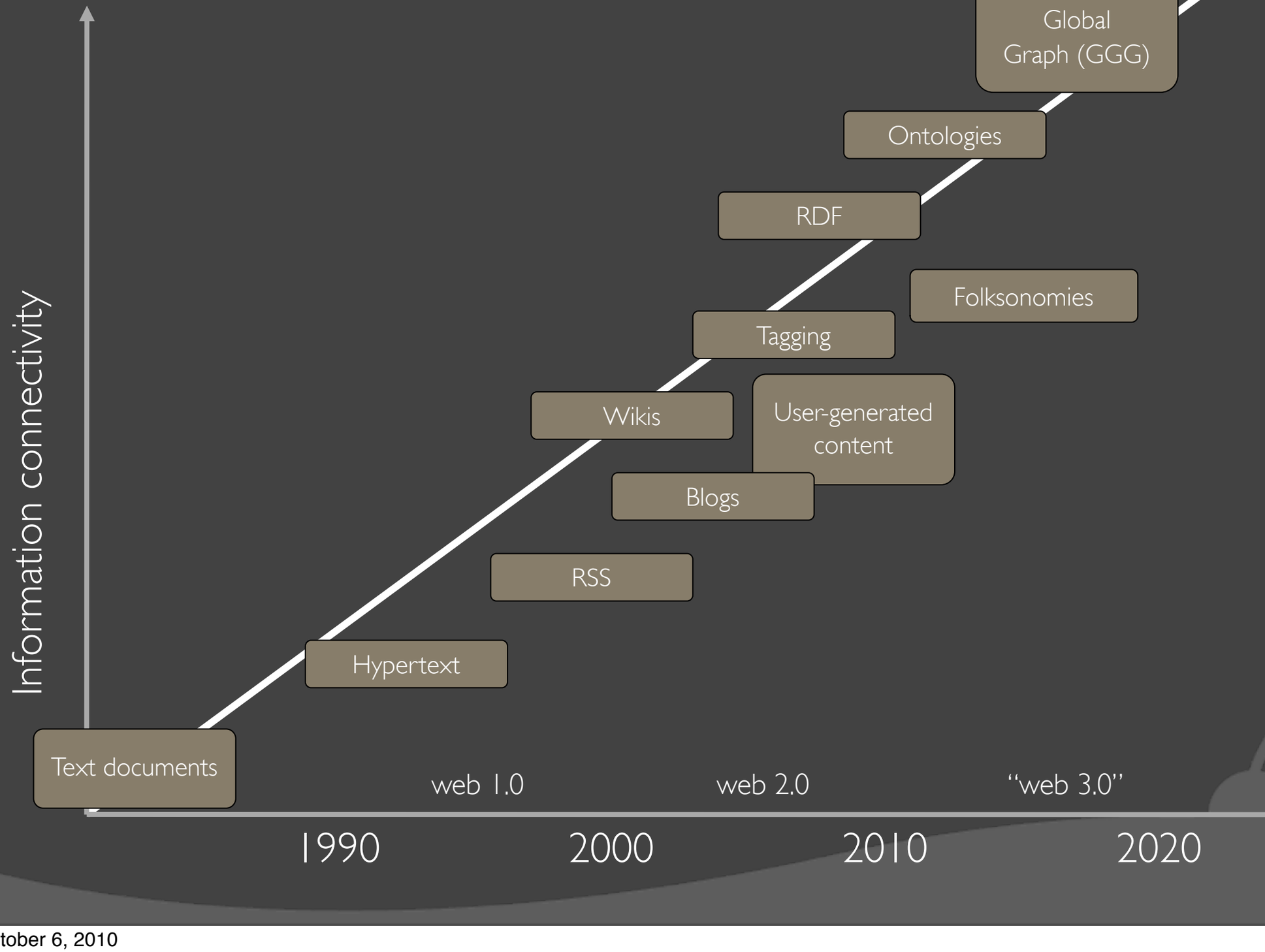
Trend 2: Connectedness



Trend 2: Connectedness



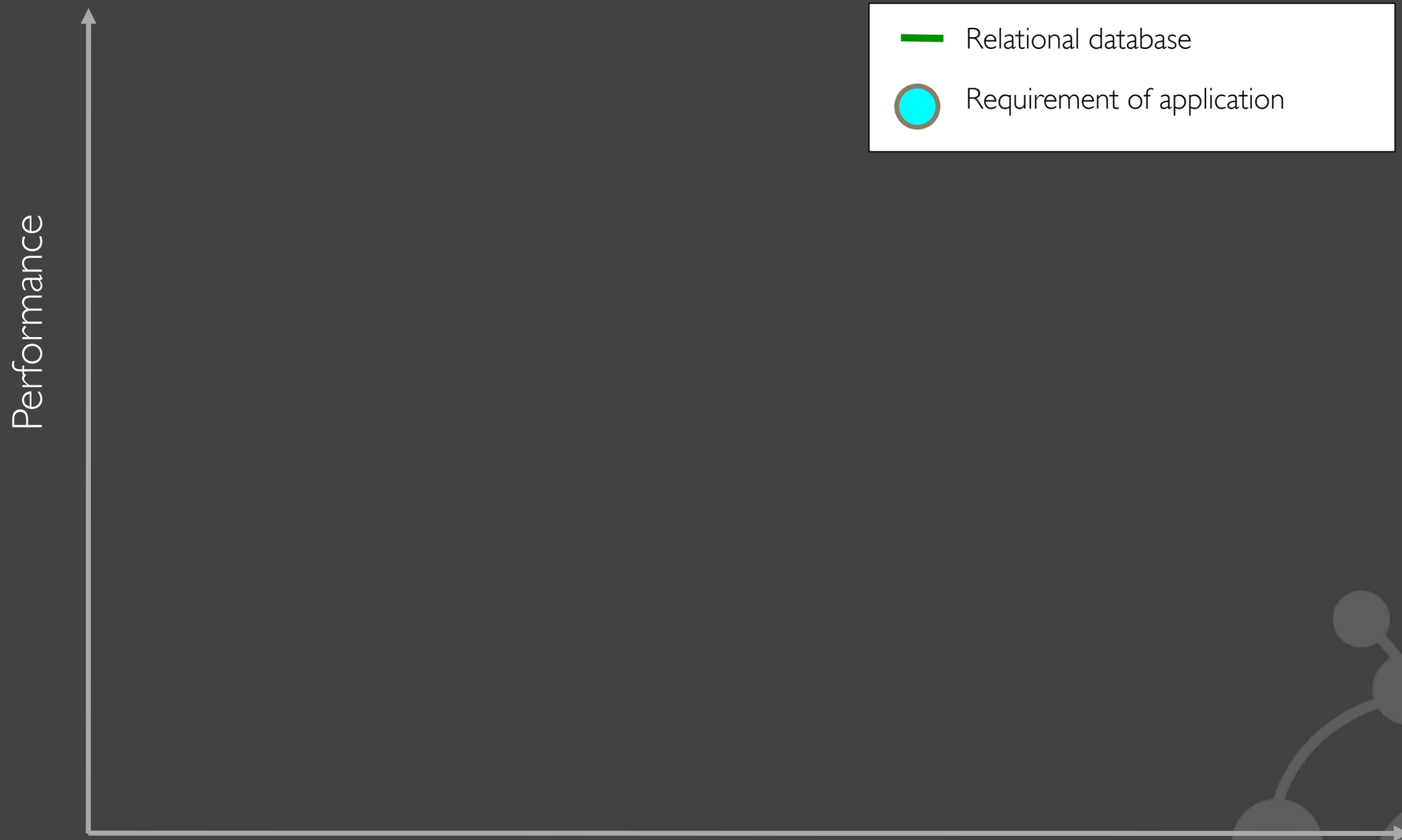
Trend 2: Connectedness



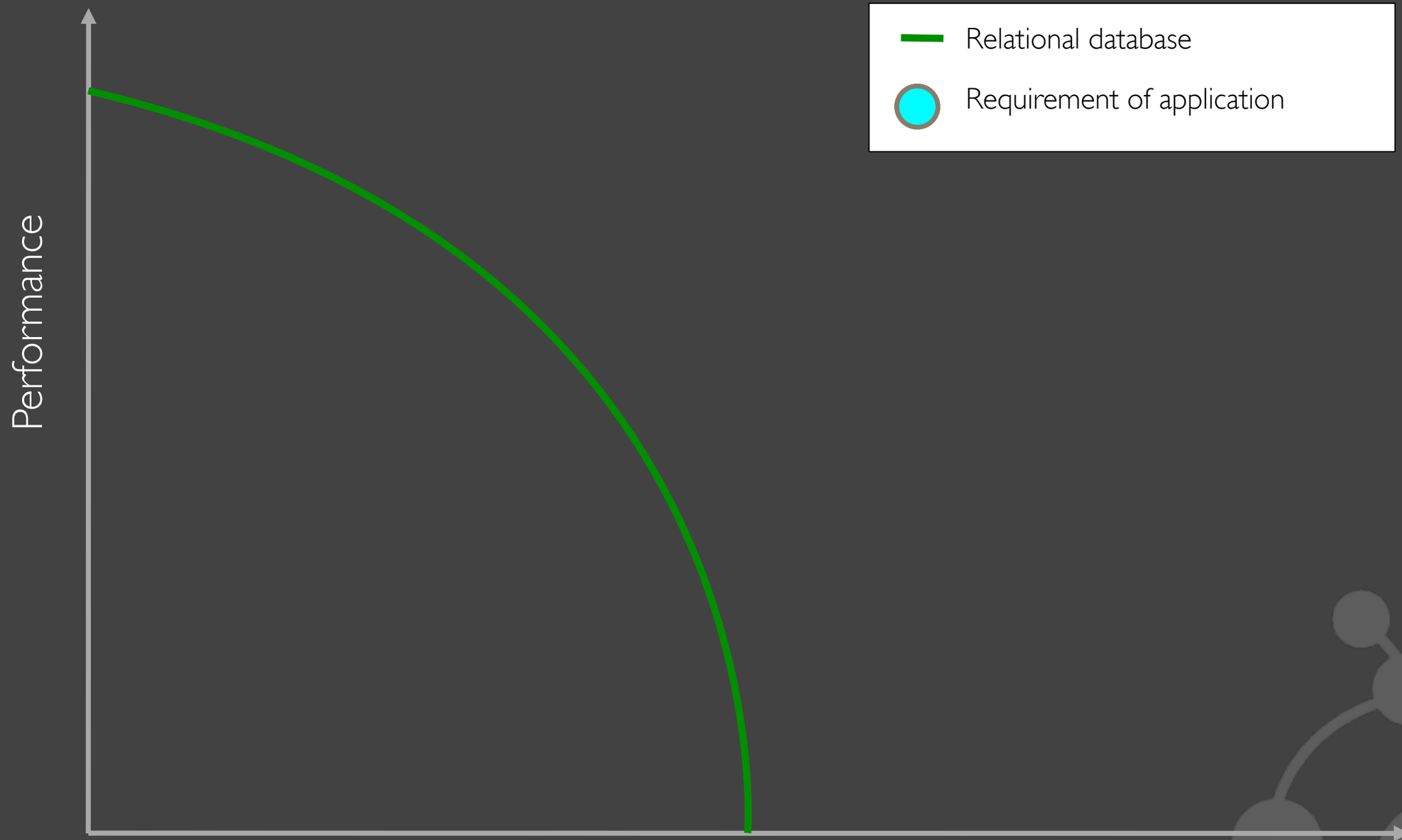
Trend 3: Semi-structure

- Individualization of content
  - In the salary lists of the 1970s, all elements had exactly one job
  - In the salary lists of the 2000s, we need 5 job columns! Or 8? Or 15?
- All encompassing "entire world views"
  - Store more data about each entity
- Trend accelerated by the decentralization of content generation that is the hallmark of the age of participation ("web 2.0")

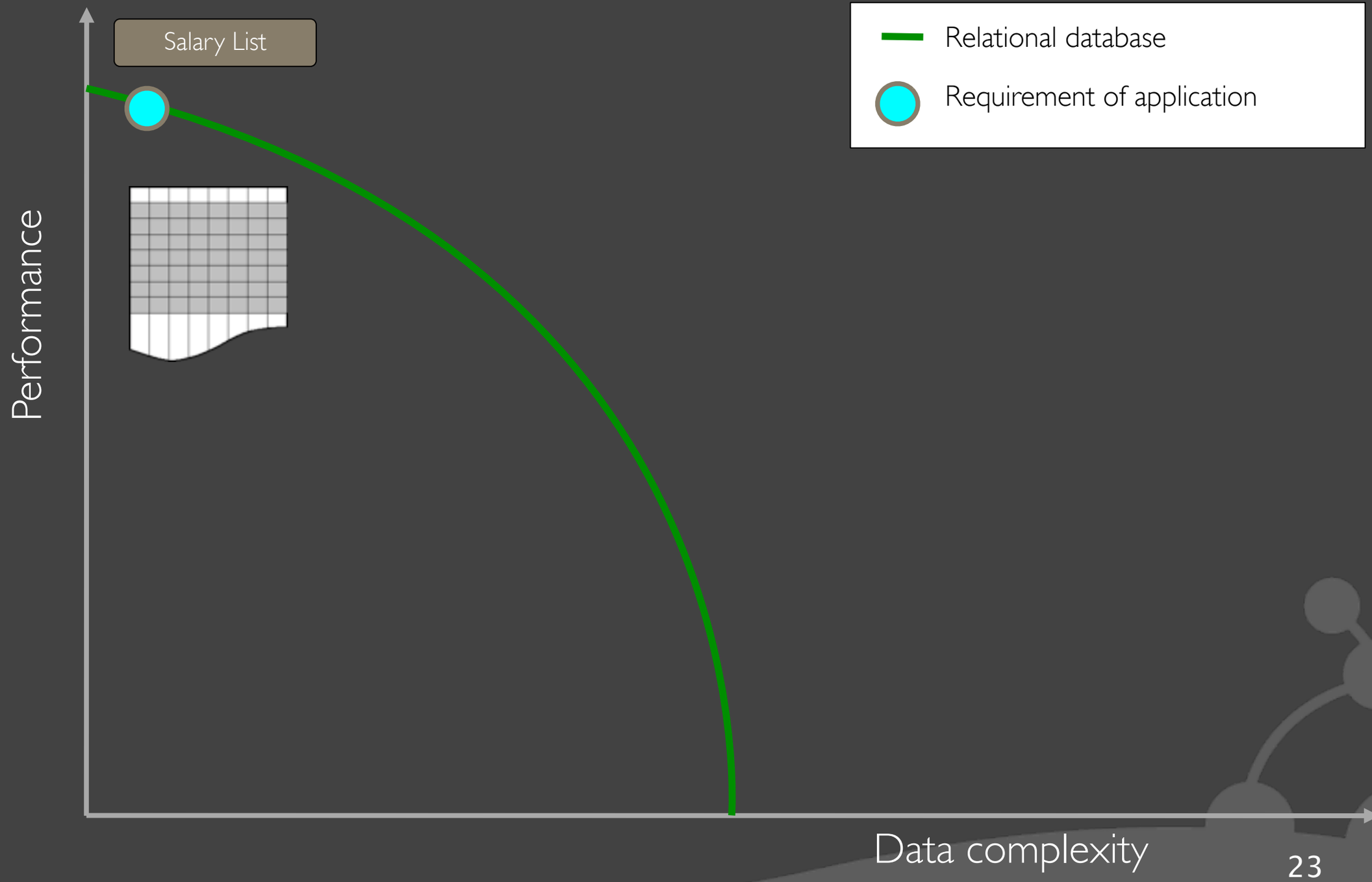
Aside: RDBMS performance



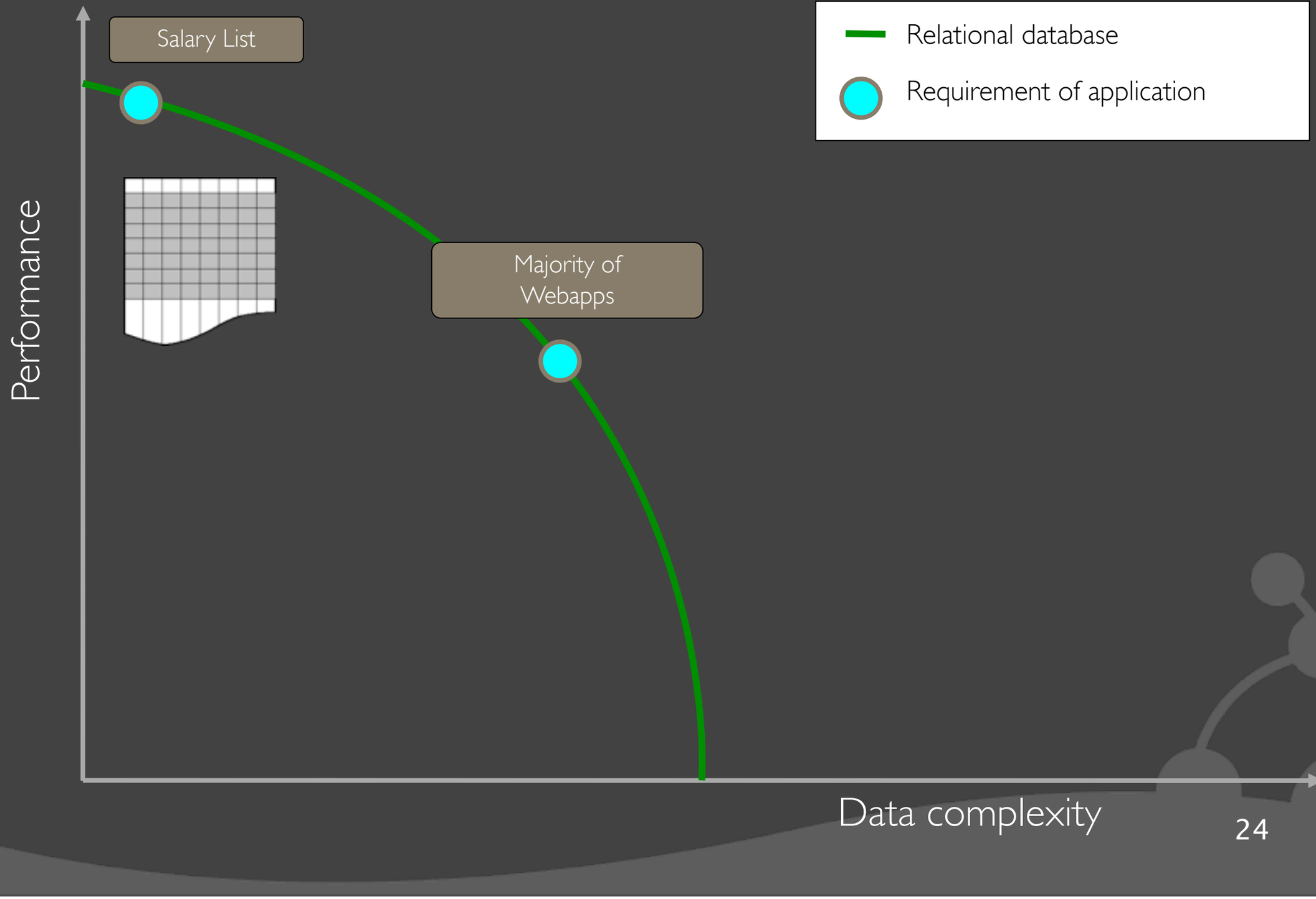
Aside: RDBMS performance



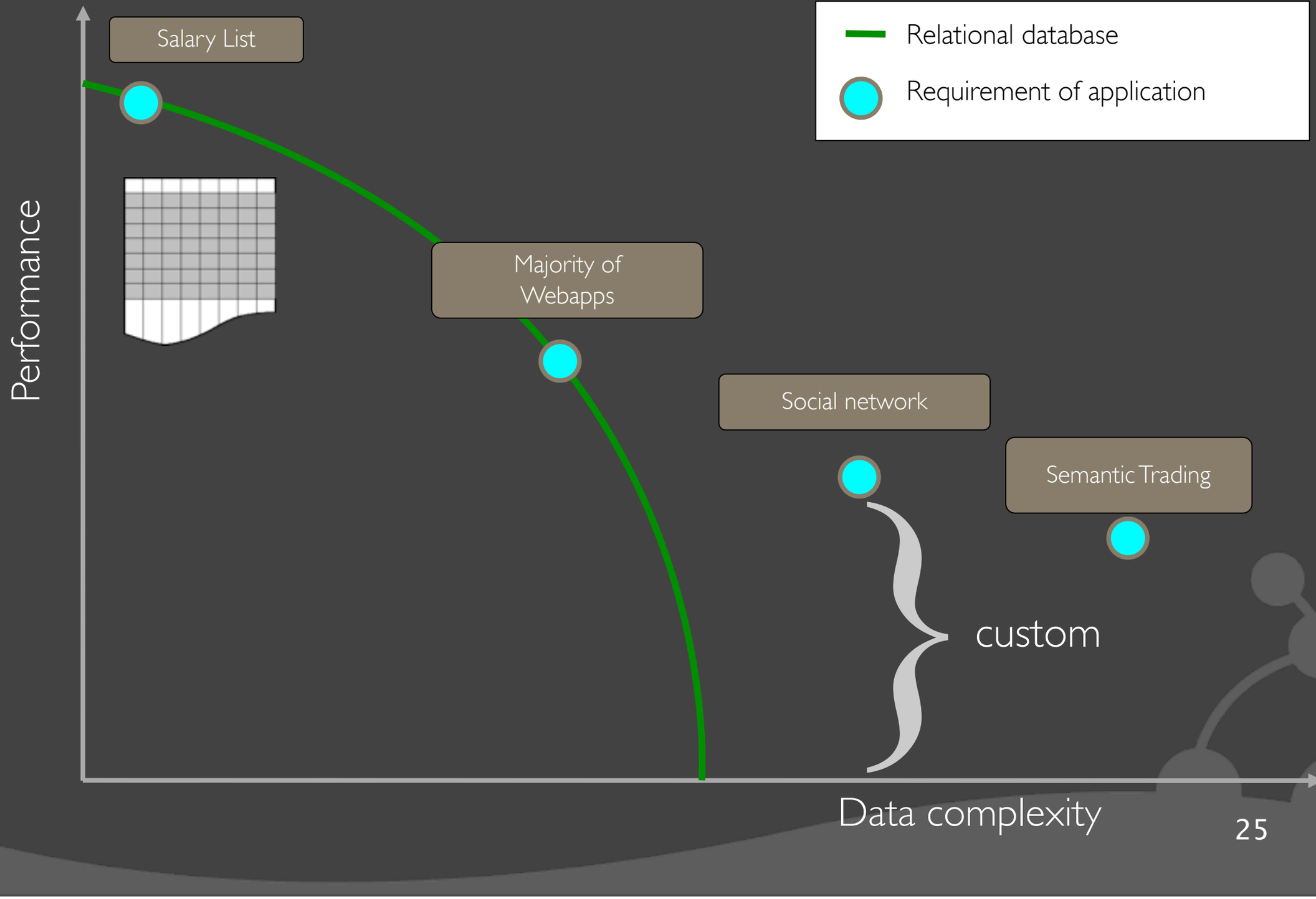
Aside: RDBMS performance



Aside: RDBMS performance

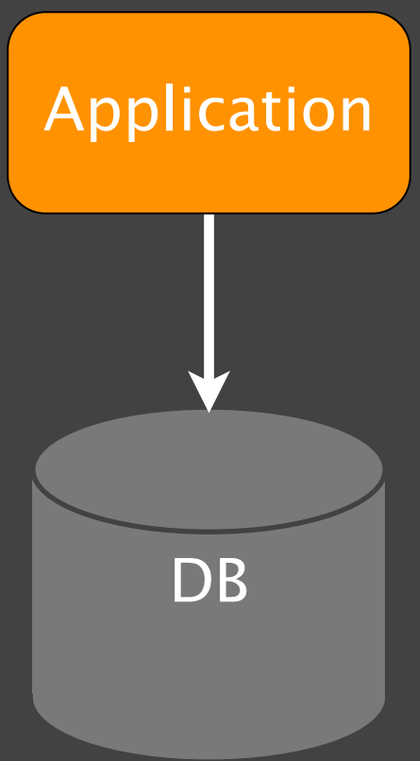


Aside: RDBMS performance



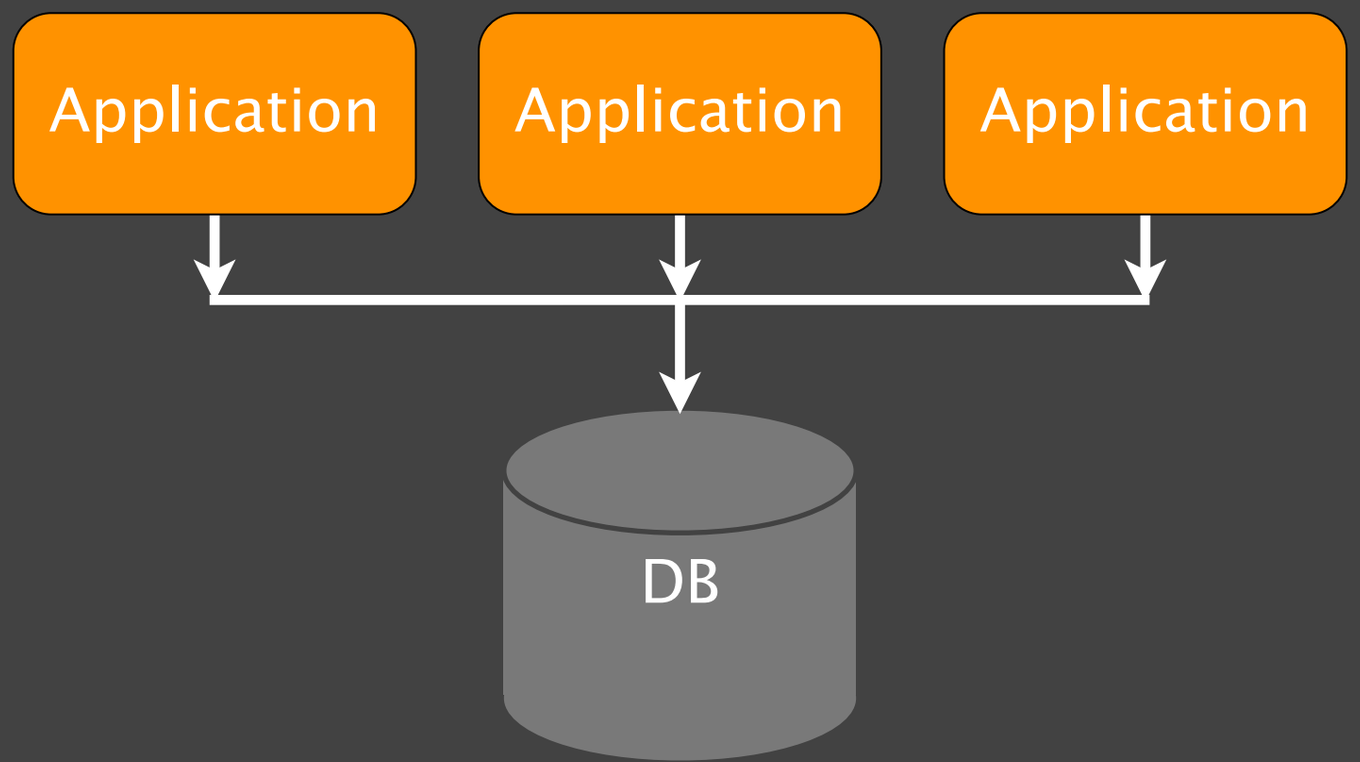
Trend 4: Architecture

1980s: Application (<-- note lack of s)

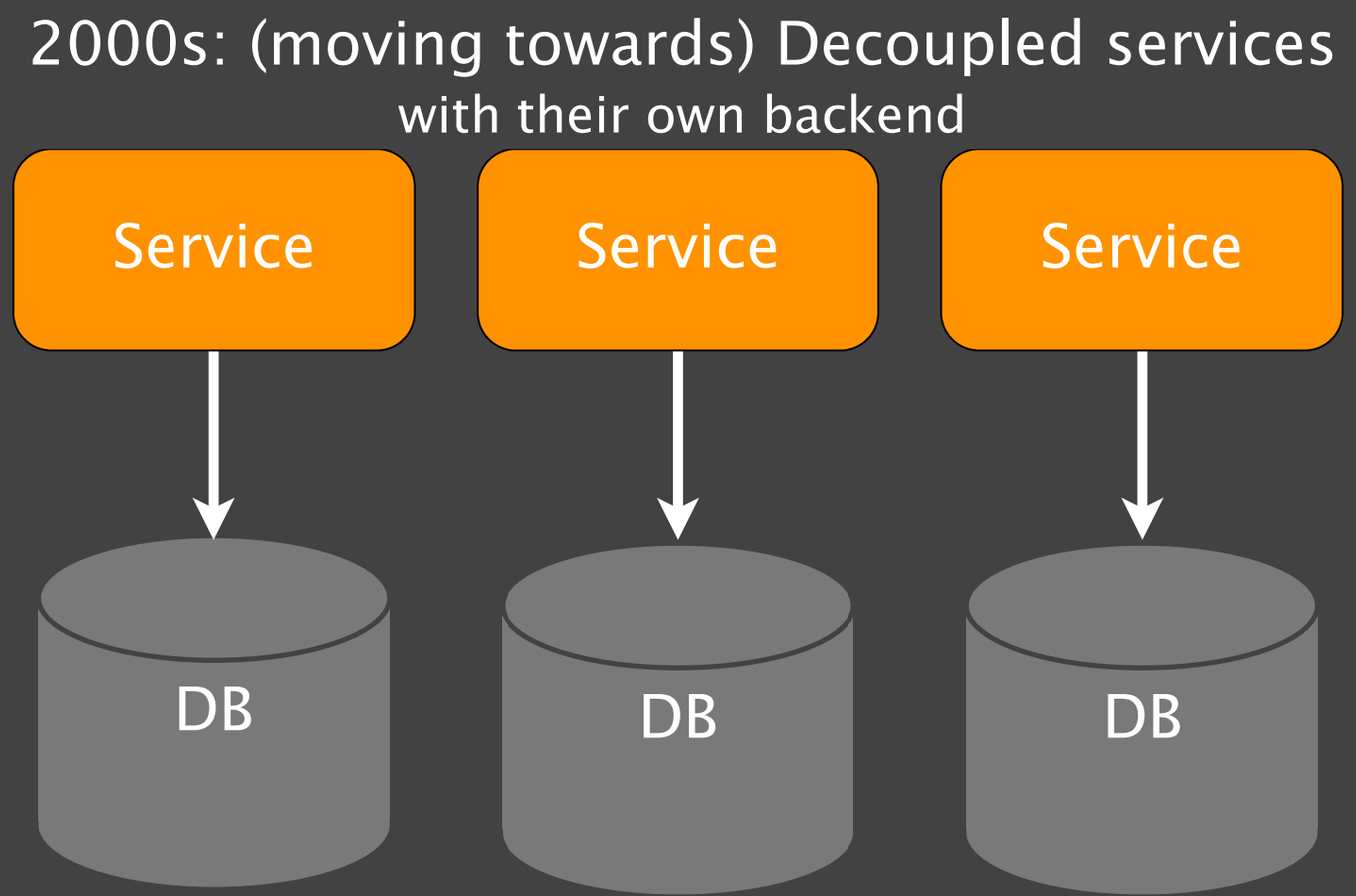


Trend 4: Architecture

1990s: Database as integration hub



Trend 4: Architecture



Why NOSQL Now?

- Trend 1: Size
- Trend 2: Connectedness
- Trend 3: Semi-structure
- Trend 4: Architecture

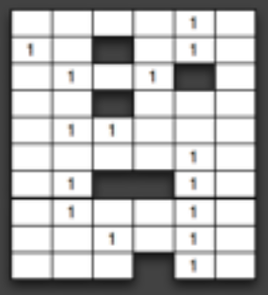
neo technology
value in relationships

Key-Value

Graph DB

Four NOSQL categories

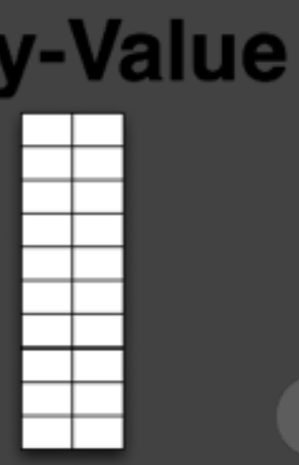
BigTable



Document

Category 1: Key-Value stores

- Lineage:
  - "Dynamo: Amazon's Highly Available Key-Value Store" (2007)
- Data model:
  - Global key-value mapping
  - Think: Globally available HashMap/Dict/etc
- Examples:
  - Project Voldemort
  - Tokyo {Cabinet, Tyrant, etc}



Category 1: Key-Value stores

- Strengths
  - Simple data model
  - Great at scaling out horizontally
- Weaknesses:
  - Simplistic data model
  - Poor for complex data

Category II: ColumnFamily (BigTable) stores

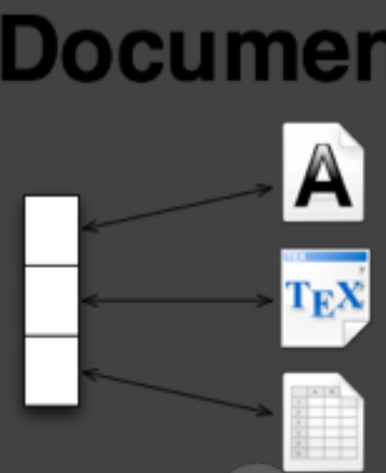
- Lineage:
  - "Bigtable: A Distributed Storage System for Structured Data" (2006)

- Data model:
  - A big table, with column families

- Examples:
  - HBase
  - HyperTable
  - Cassandra

Category III: Document databases

- Lineage:
  - Lotus Notes
- Data model:
  - Collections of documents
  - A document is a key-value collection
- Examples:
  - CouchDB
  - MongoDB



Document db: An example

- How would we model a blogging software?

- One stab:
  - Represent each Blog as a *Collection* of Post *documents*
  - Represent Comments as *nested documents* in the Post documents

Document db: Creating a blog post

```
import com.mongodb.Mongo;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;
// ...
Mongo mongo = new Mongo( "localhost" ); // Connect to MongoDB
// ...
DB blogs = mongo.getDB( "blogs" ); // Access the blogs database
DBCollection myBlog = blogs.getCollection( "Thobe's blog" );

DBObject blogPost = new BasicDBObject();
blogPost.put( "title", "JAOO^H^H^H^HGoto; con 2010" );
blogPost.put( "pub_date", new Date() );
blogPost.put( "body", "Publishing a post about JA...Goto con in
   my MongoDB blog!" );
blogPost.put( "tags", Arrays.asList( "conference", "names" ) );
blogPost.put( "comments", new ArrayList() );

myBlog.insert( blogPost );
```

Retrieving posts

```
// ...
import com.mongodb.DBCursor;
// ...

public Object getAllPosts( String blogName ) {
    DBCollection blog = db.getCollection( blogName );
    return renderPosts( blog.find() );
}

private Object renderPosts( DBCursor cursor ) {
    // order by publication date (descending)
    cursor = cursor.sort( new BasicDBObject( "pub_date", -1 ) );
    // ...
}
```

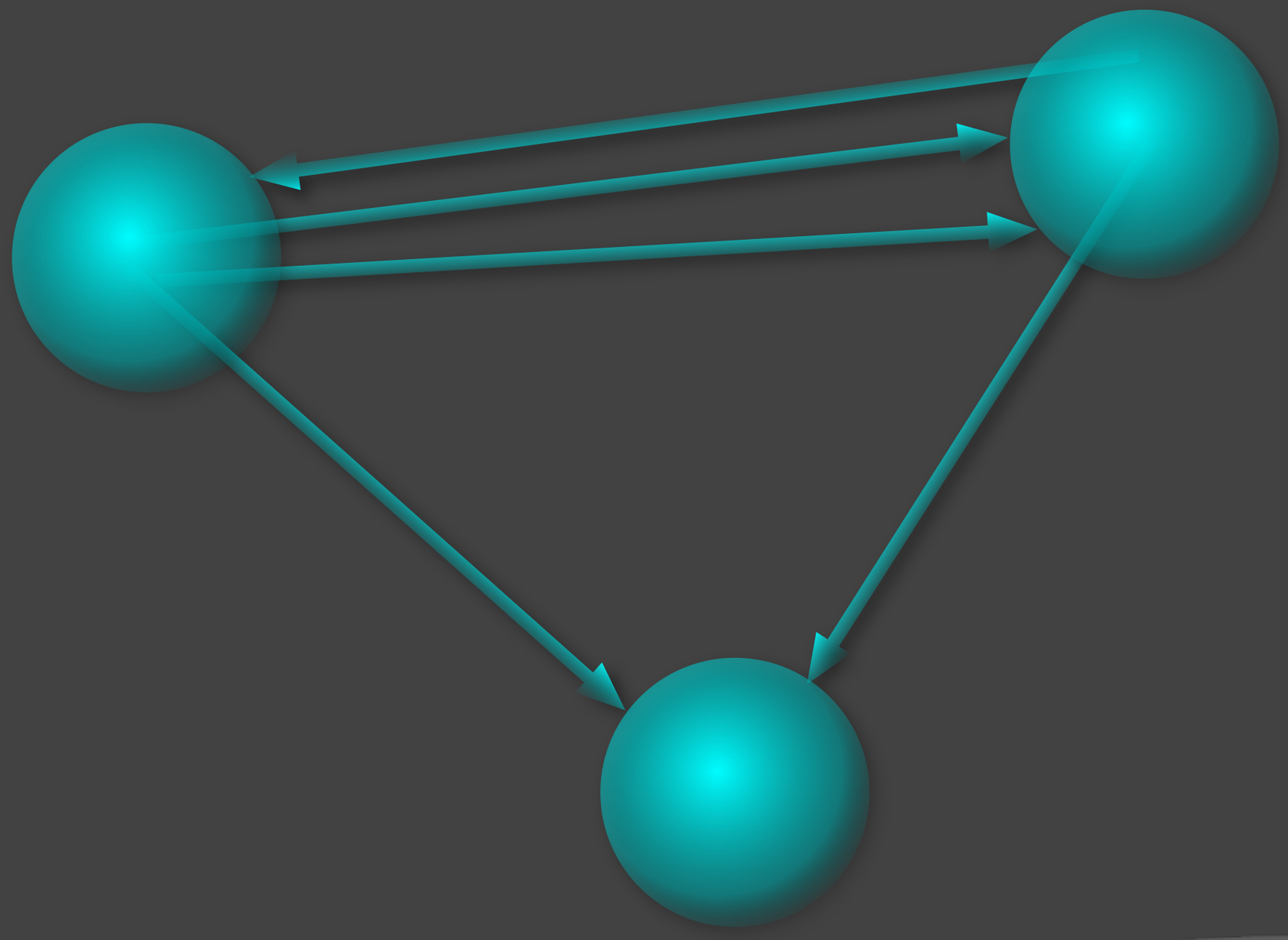
Category IV: Graph databases

- Lineage:
  - Euler and graph theory

- Data model:
  - Nodes with properties
  - Typed relationships with properties

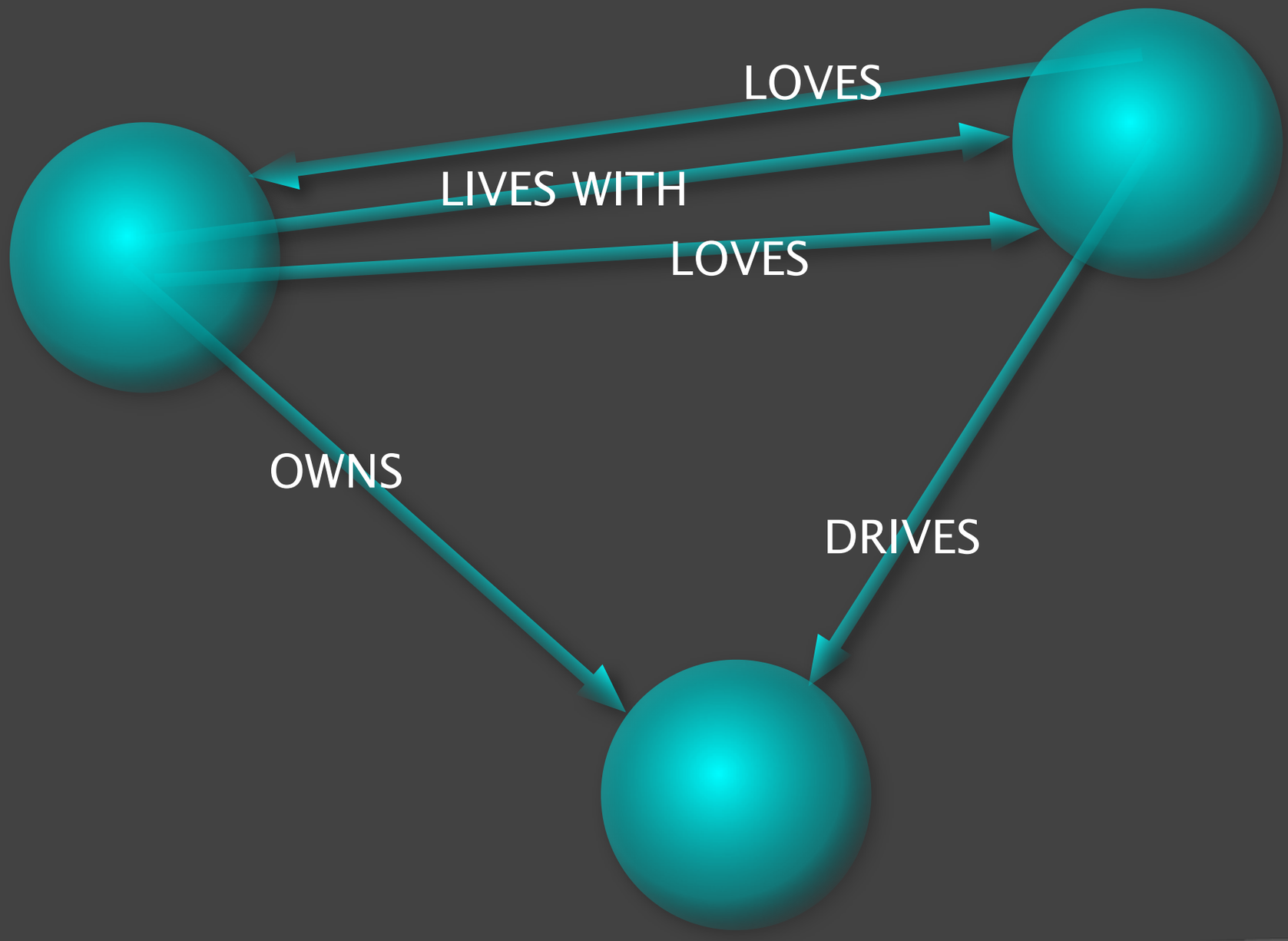
- Examples:
  - Sones GraphDB
  - InfiniteGraph
  - Neo4j

Property Graph model



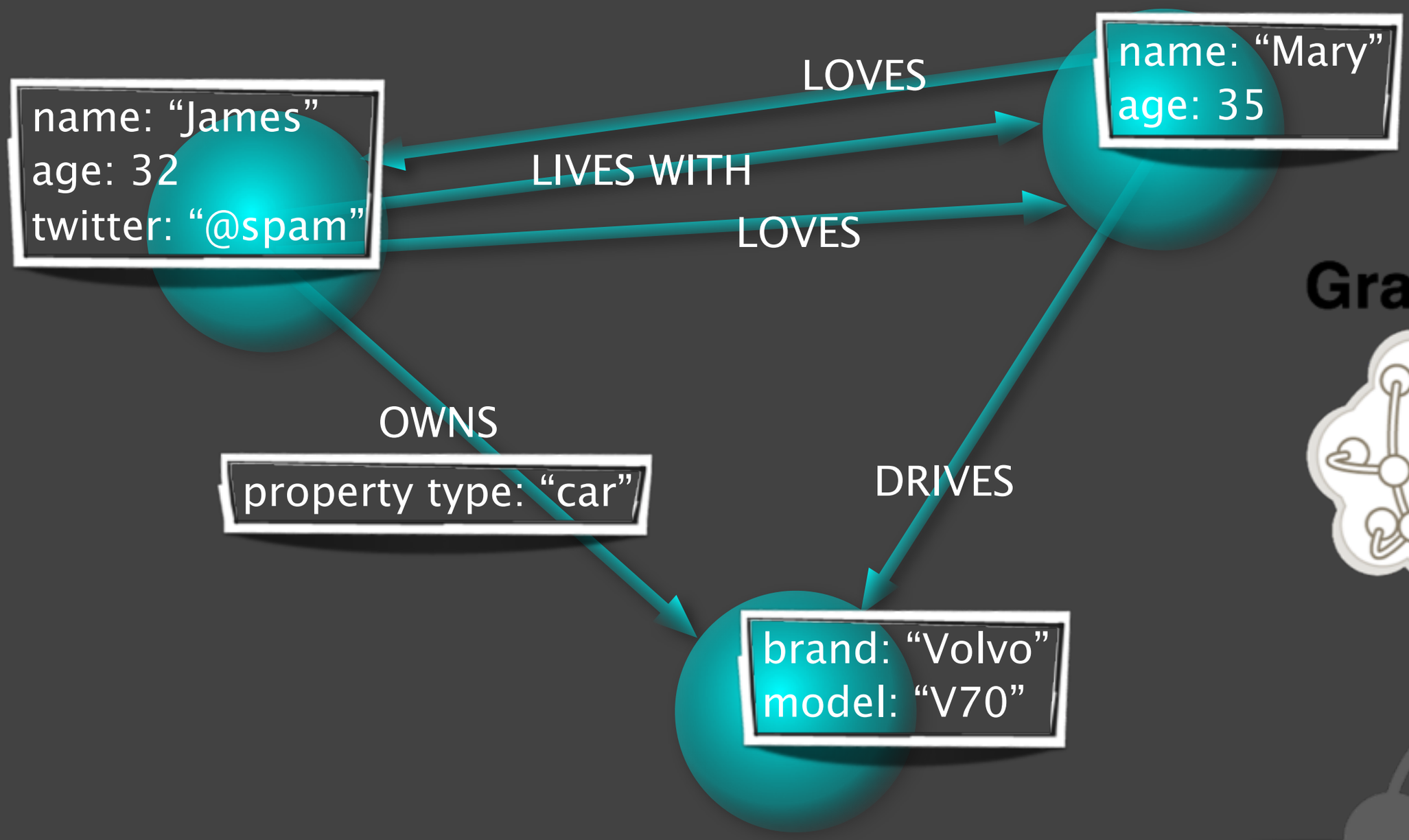
Graph DB

Property Graph model



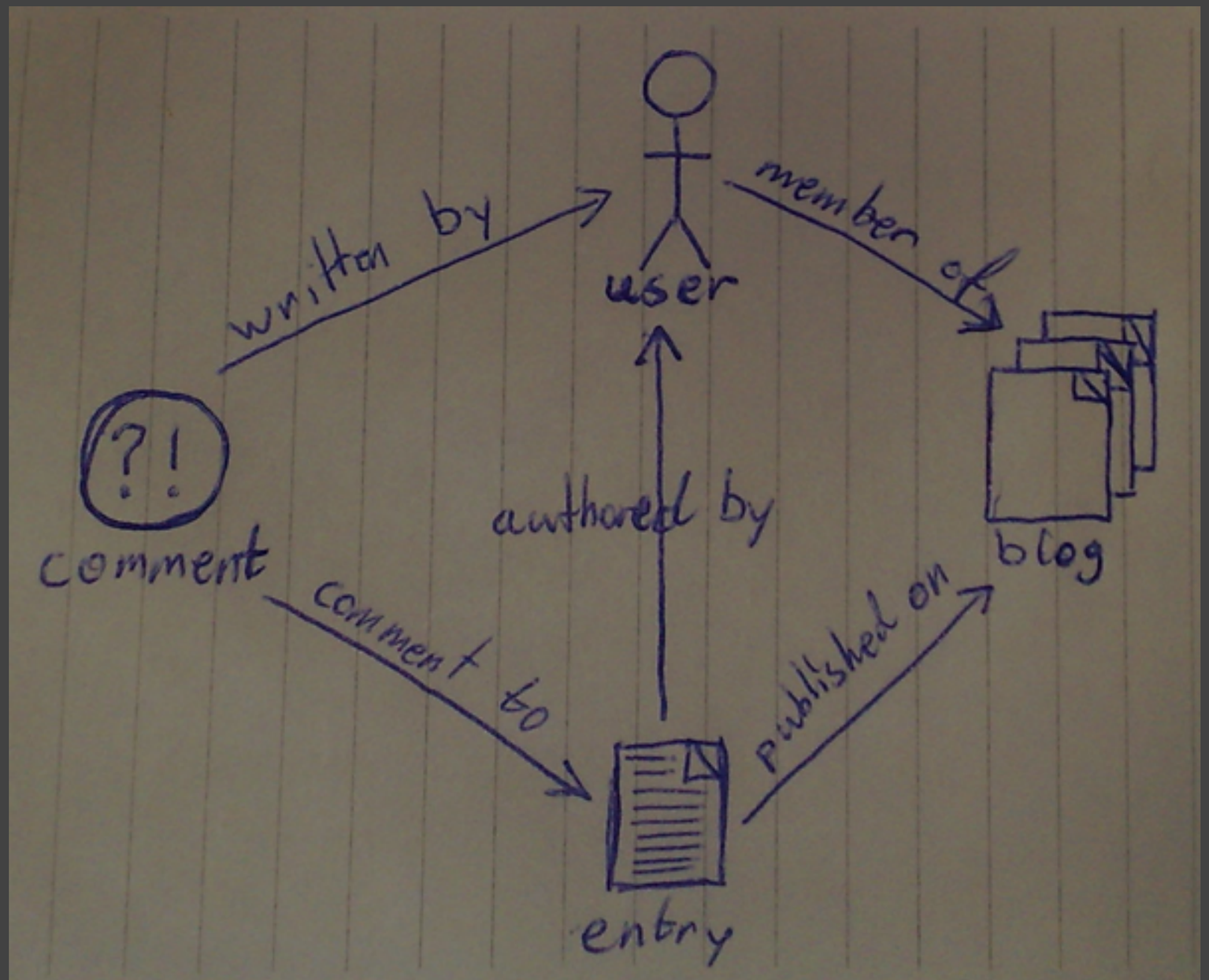
Graph DB

Property Graph model



Graph DB

Graphs are whiteboard friendly

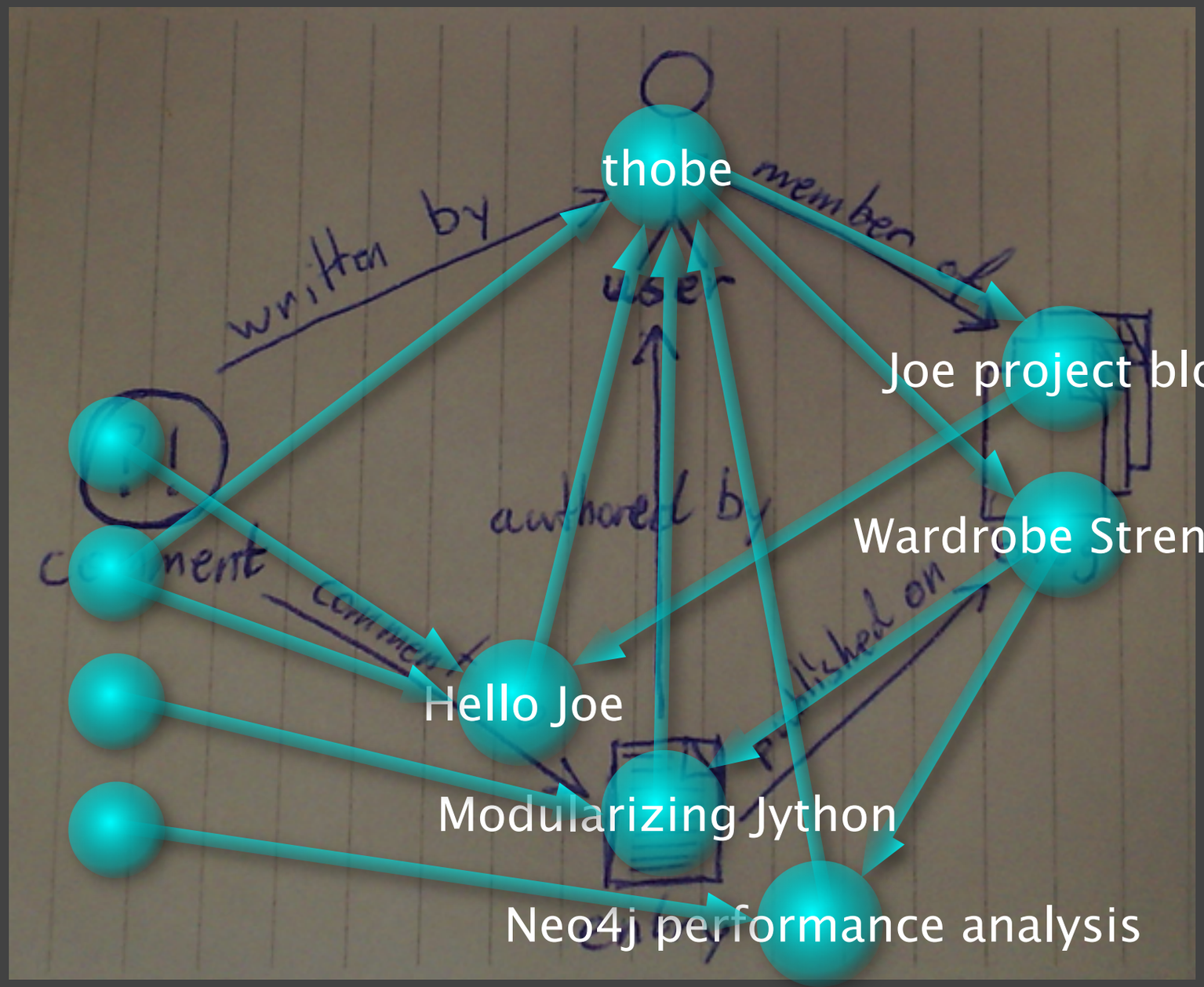


An application domain model outlined on a whiteboard or piece of paper would be translated to an ER-diagram, then normalized to fit a Relational Database. With a Graph Database the model from the whiteboard is implemented directly.

Graph DB

Image credits: Tobias Ivarsson

Graphs are whiteboard friendly



An application domain model outlined on a whiteboard or piece of paper would be translated to an ER-diagram, then normalized to fit a Relational Database. With a Graph Database the model from the whiteboard is implemented directly.

Graph DB

Image credits: Tobias Ivarsson

Graph db: Creating a social graph

```
GraphDatabaseService graphDb = new EmbeddedGraphDatabase(
   GRAPH_STORAGE_LOCATION );

Transaction tx = graphDb.beginTx();
try {
    Node mrAnderson = graphDb.createNode();
    mrAnderson.setProperty( "name", "Thomas Anderson" );
    mrAnderson.setProperty( "age", 29 );

    Node morpheus = graphDb.createNode();
    morpheus.setProperty( "name", "Morpheus" );
    morpheus.setProperty( "rank", "Captain" );

    Relationship friendship = mrAnderson.createRelationshipTo(
      morpheus, SocialGraphTypes.FRIENDSHIP );

    tx.success();
} finally {
    tx.finish();
}
```

Graph db: How do I know this person?

```
Node me = ...
Node you = ...

PathFinder shortestPathFinder = GraphAlgoFactory.shortestPath(
    Traversals.expanderForTypes(
        SocialGraphTypes.FRIENDSHIP, Direction.BOTH ),
    /* maximum depth: */ 4 );
```

```
Path shortestPath = shortestPathFinder.findSinglePath(me, you);

for ( Node friend : shortestPath.nodes() ) {
    System.out.println( friend.getProperty( "name" ) );
}
```

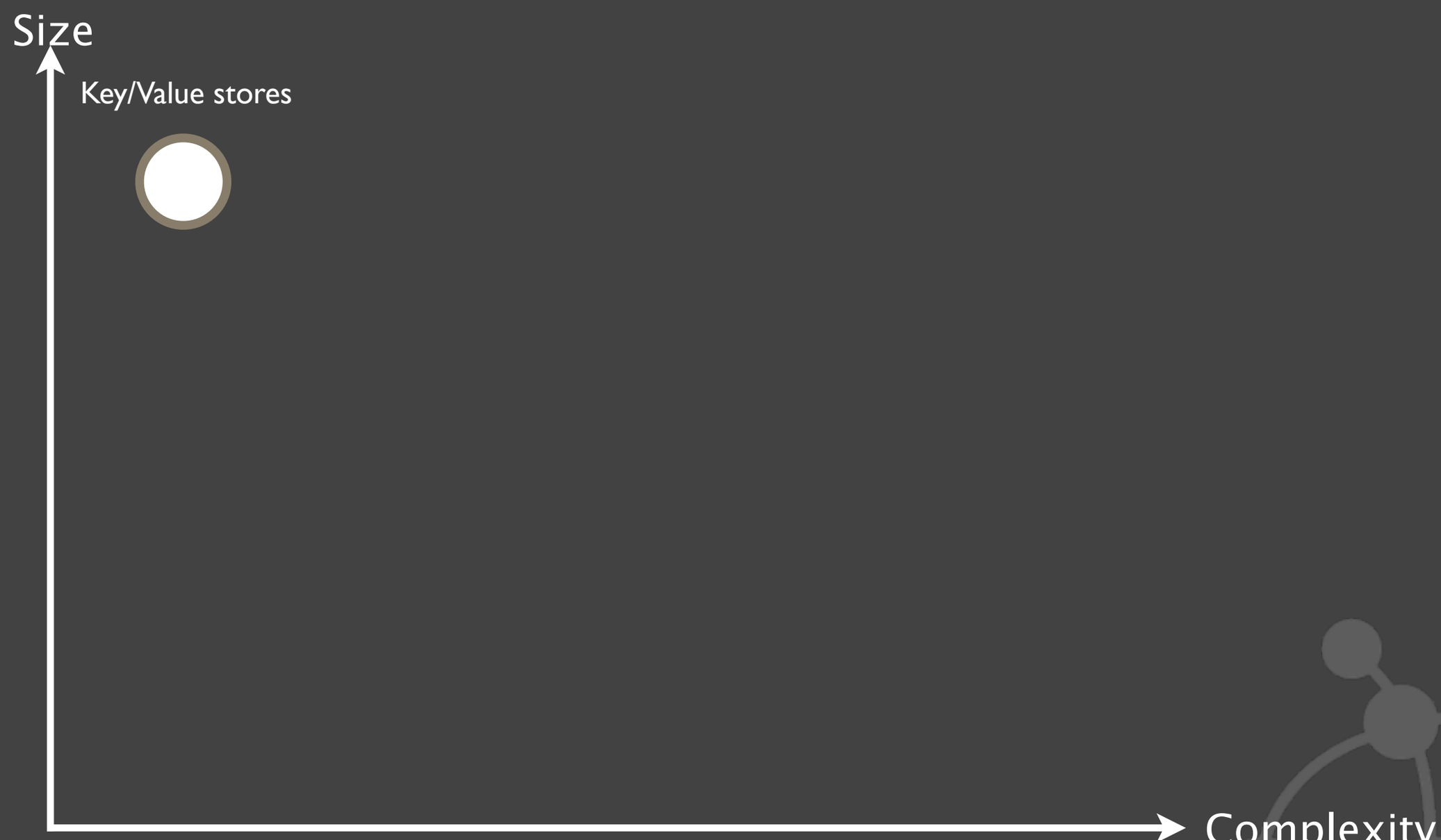
Graph db: Recommend new friends

```
Node person = ...

TraversalDescription friendsOfFriends = Traversal.description()
    .expand( Traversals.expanderForTypes(
                    SocialGraphTypes.FRIENDSHIP, Direction.BOTH ) )
    .prune( Traversal.pruneAfterDepth( 2 ) )
    .breadthFirst() // Visit my friends before their friends.
    //Visit a node at most once (don't recommend direct friends)
    .uniqueness( Uniqueness.NODE_GLOBAL )
    .filter( new Predicate<Path>() {
        // Only return friends of friends
        public boolean accept( Path traversalPos ) {
            return traversalPos.length() == 2;
        }
    } );

for ( Node recommendation :
            friendsOfFriends.traverse( person ).nodes() ) {
    System.out.println( recommendedFriend.getProperty("name") );
}
```

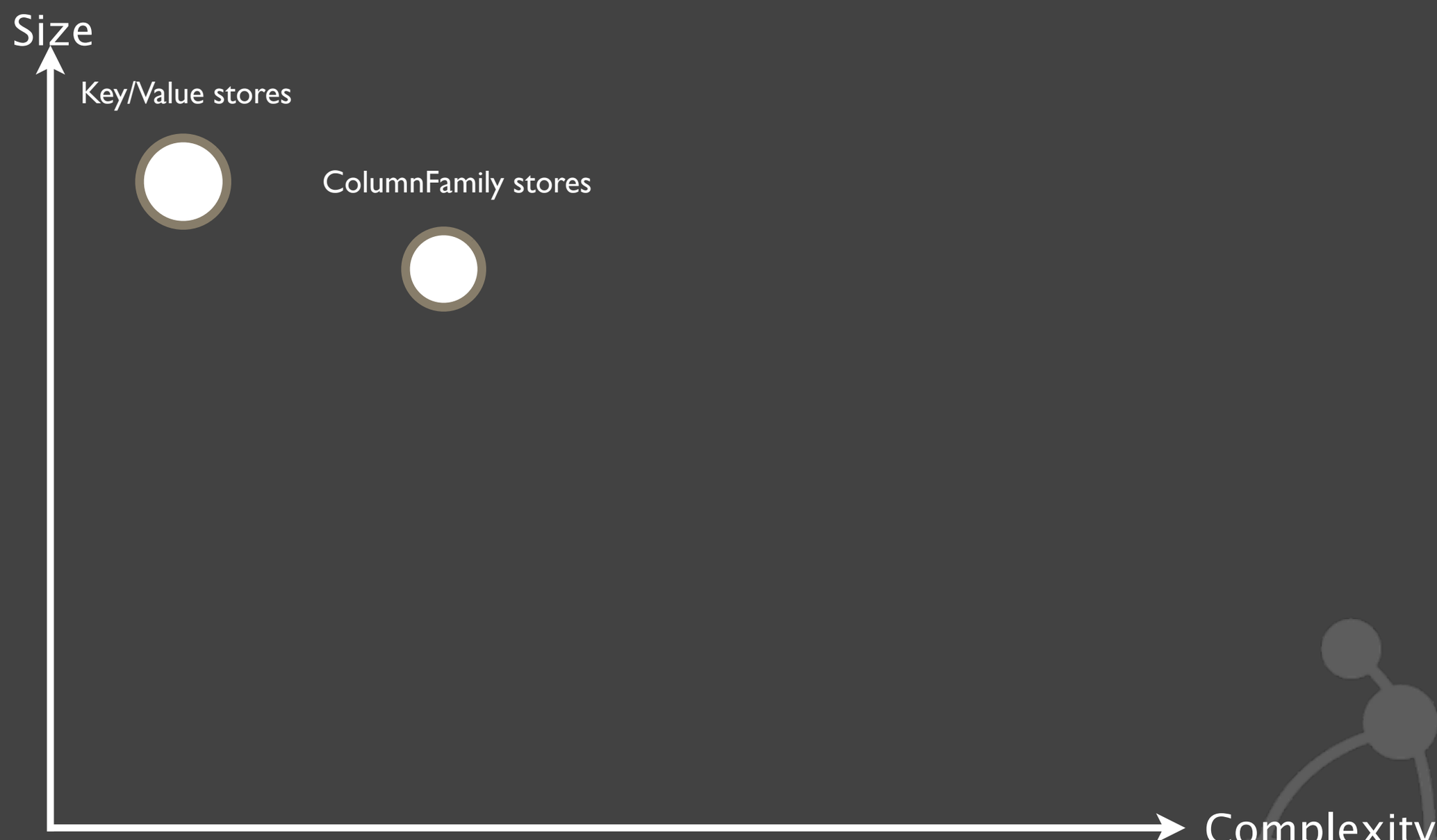
Four emerging NOSQL categories

- Key-Value stores
- ColumnFamiy stores
- Document databases
- Graph databases

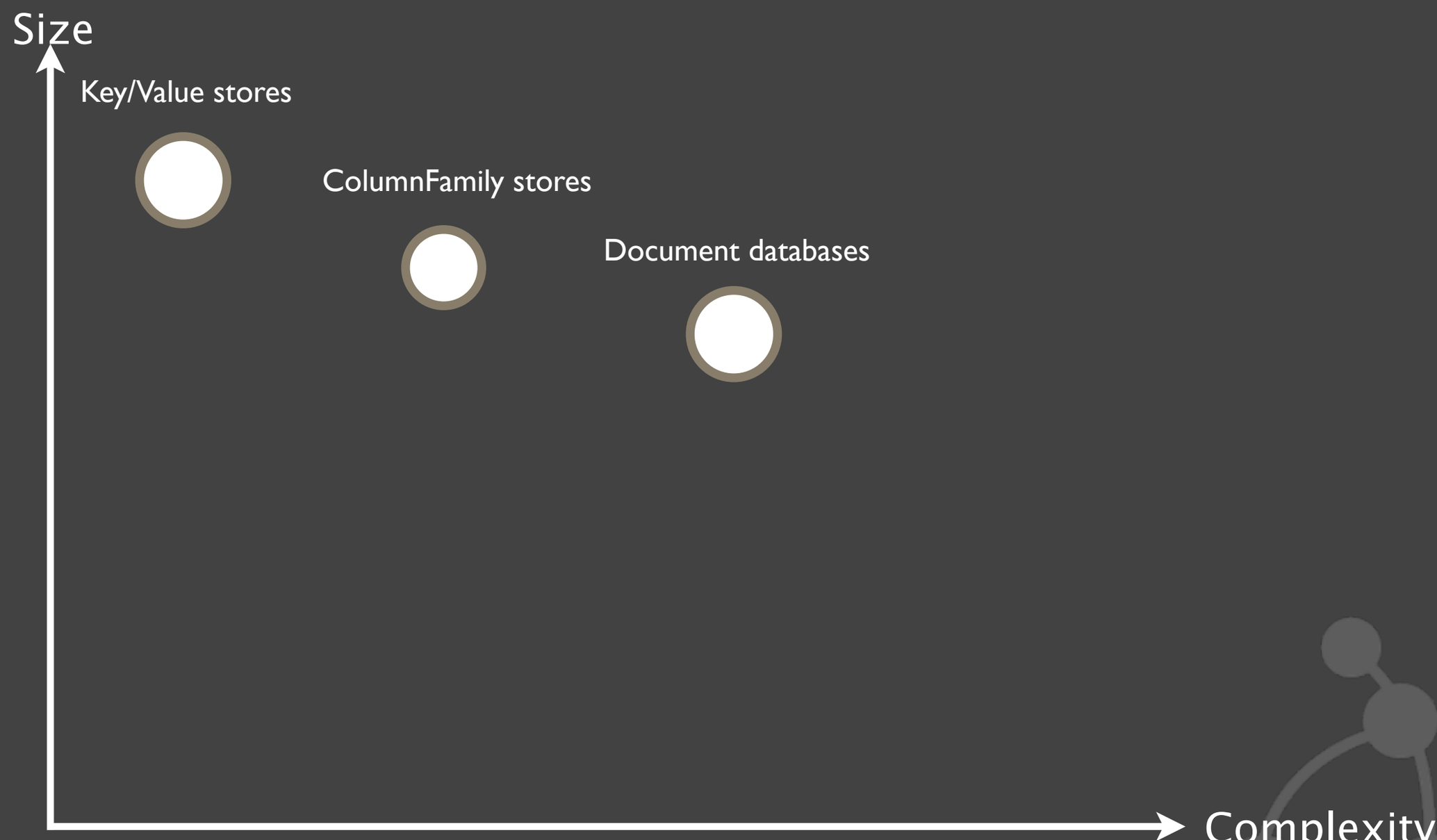
Scaling to size vs. Scaling to complexity



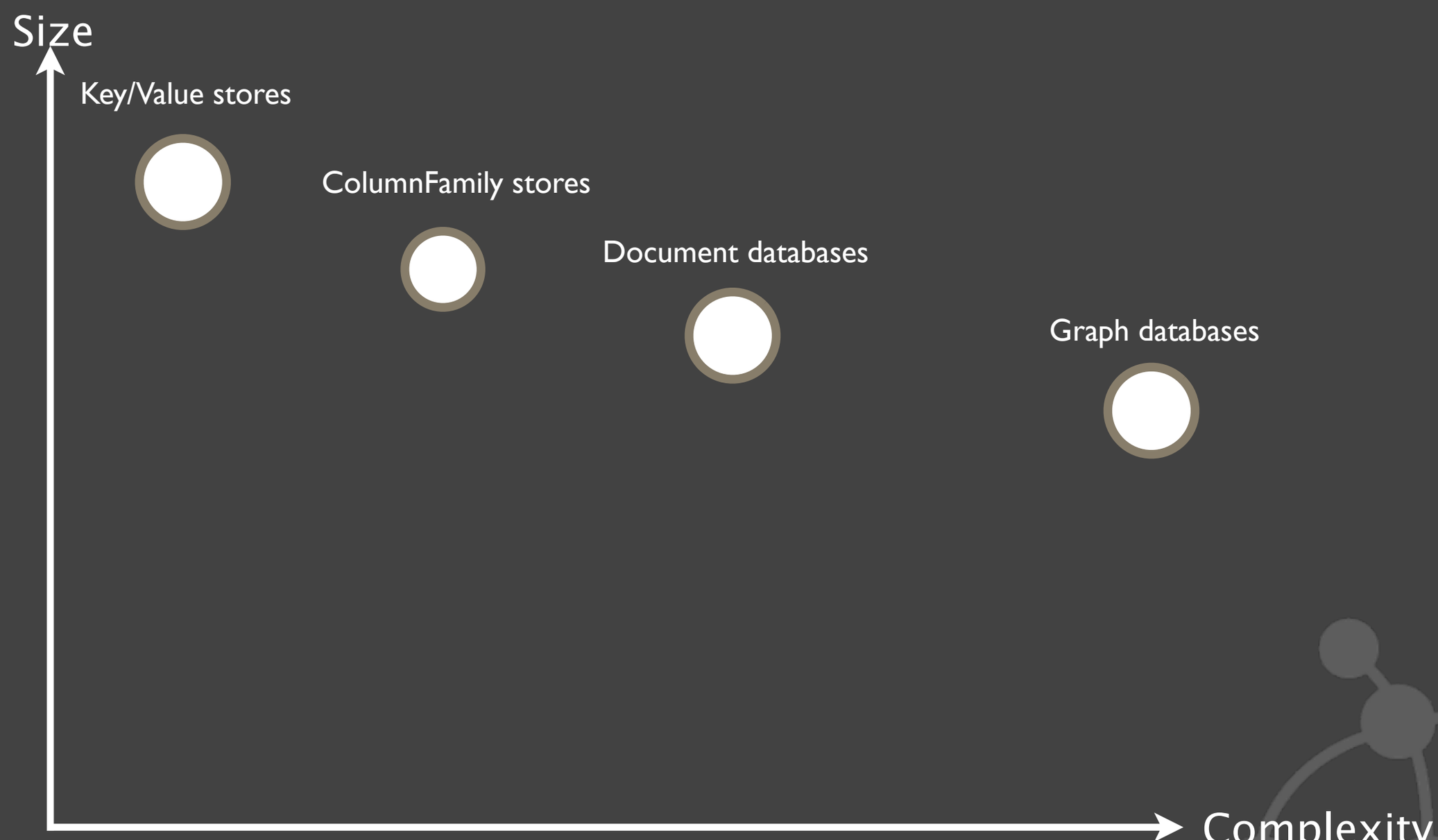
Scaling to size vs. Scaling to complexity



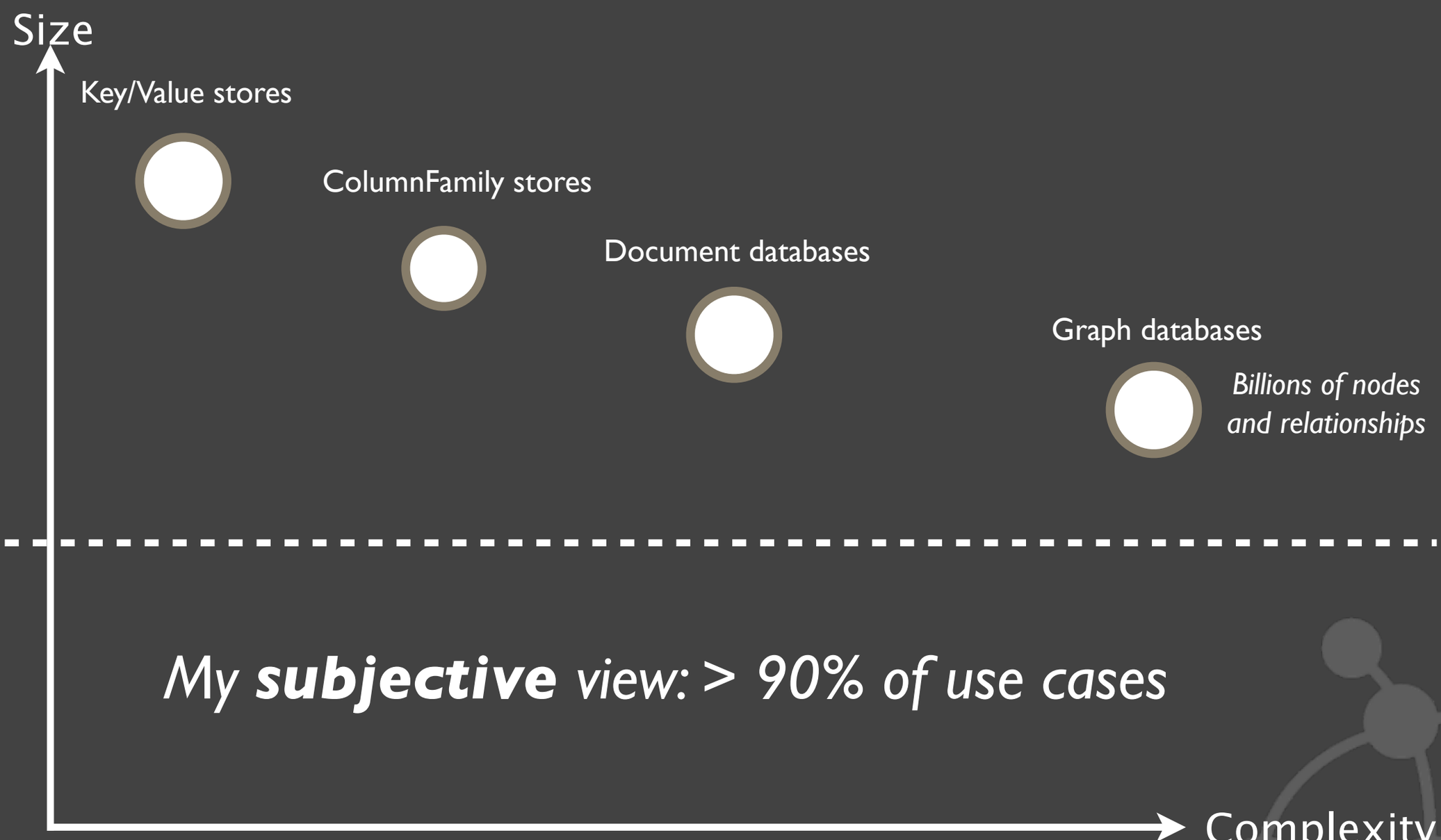
Scaling to size vs. Scaling to complexity



Scaling to size vs. Scaling to complexity



Scaling to size vs. Scaling to complexity



NOSQL challenges?

NOSQL challenges?

- Mindshare
  - But that's also product usability ("how do you query it?")

NOSQL challenges?

- Mindshare
  - But that's also product usability ("how do you query it?")
- Tool support
  - Both devtime tools and runtime ops tools
  - Standards may help?
  - ... or maybe just time

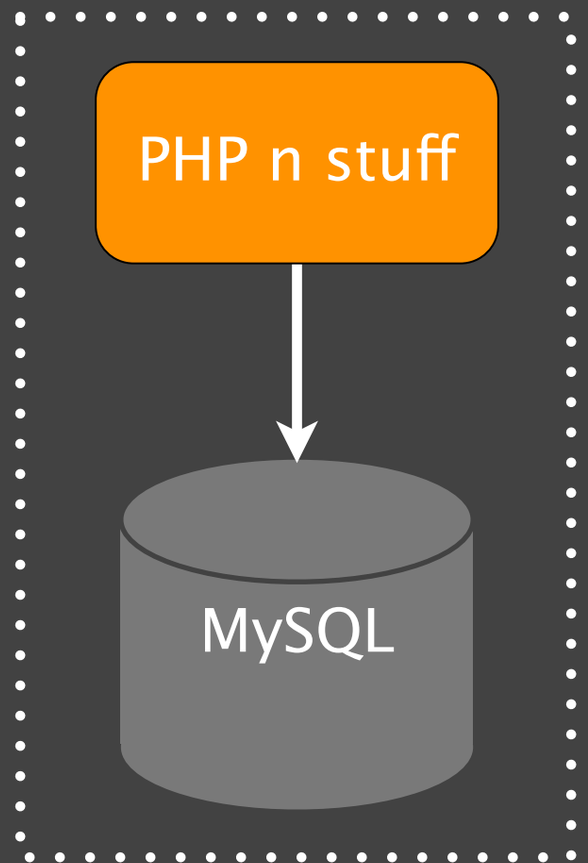
NOSQL challenges?

- Mindshare
  - But that's also product usability ("how do you query it?")
- Tool support
  - Both devtime tools and runtime ops tools
  - Standards may help?
  - ... or maybe just time
- Middleware support

Middleware support?

- Let me tell you the story about Mike

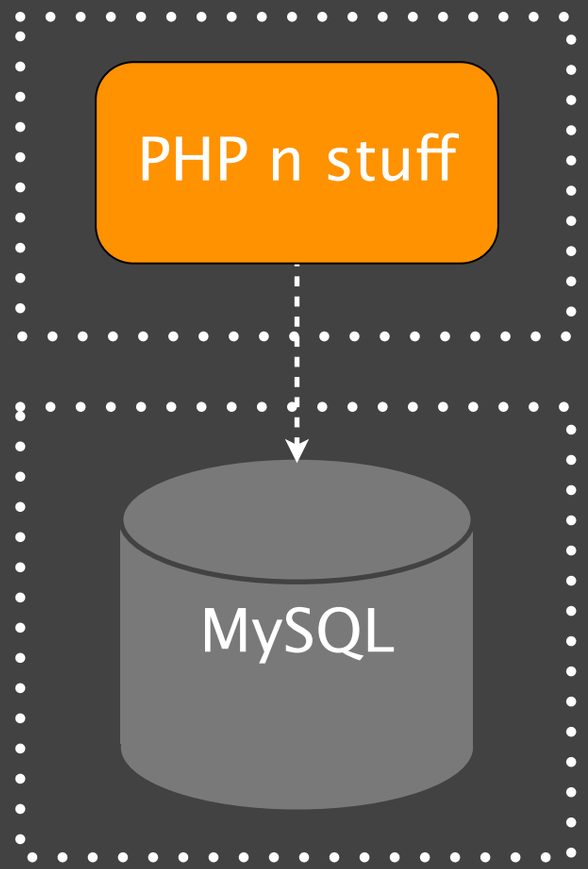
Step 1: Buildsing a web site



One box

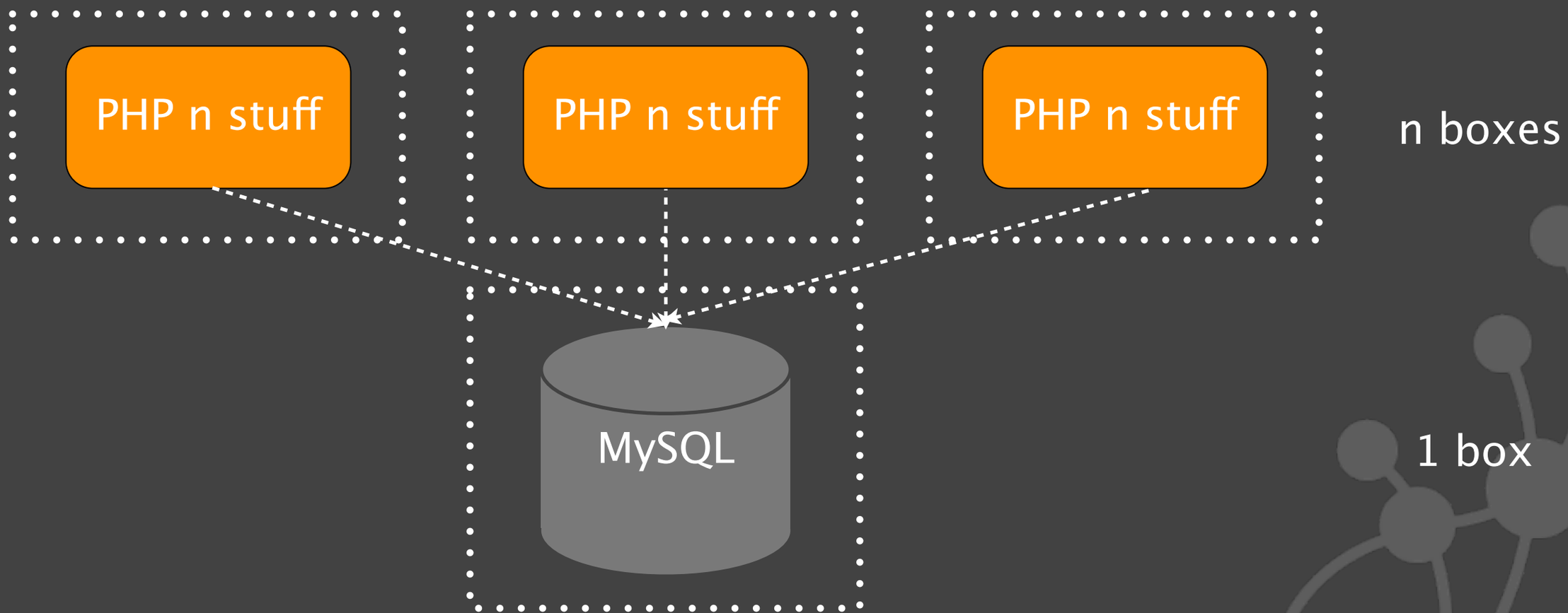
Step II: Whoa, ppl are actually using it?

Step II: Whoa, ppl are actually using it?

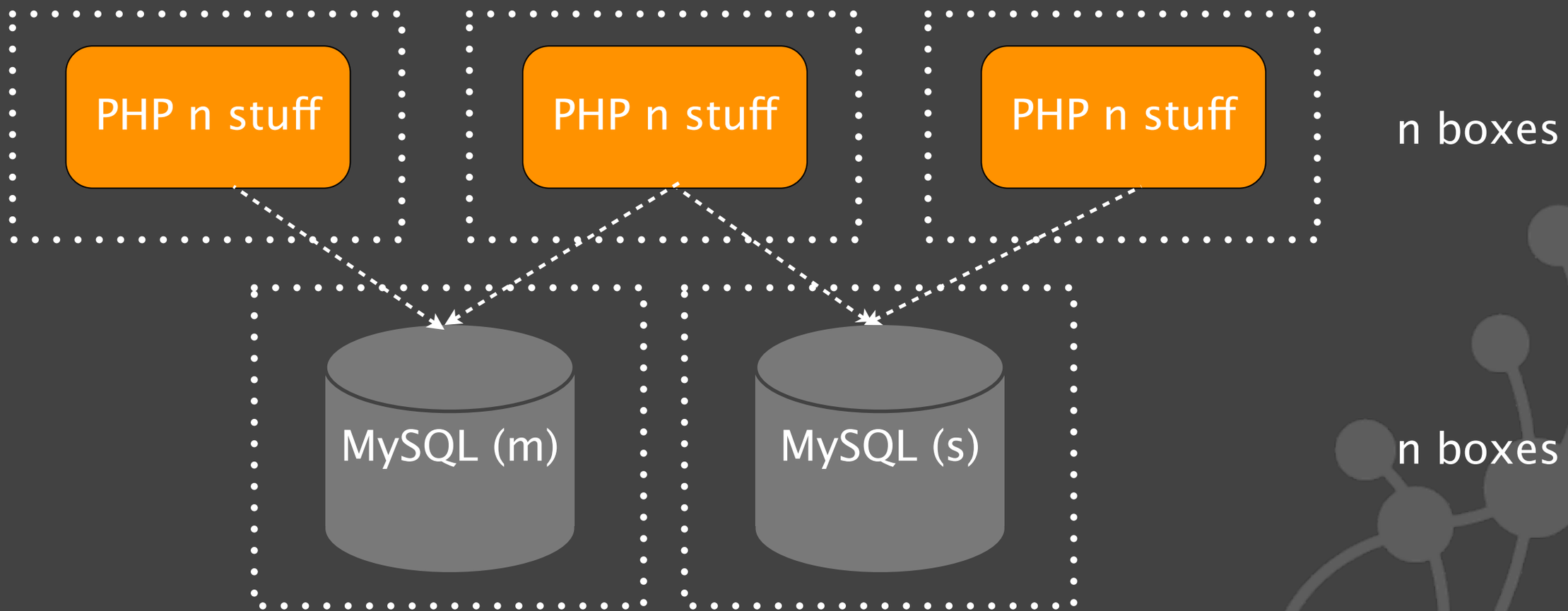


Two boxes

Step III: That's a LOT of pages served...



Step IV: Our DB is completely overwhelmed...



Step V: Our DBs are STILL overwhelmed

?

Step V: Our DBs are STILL overwhelmed

- Turns out the problem is due to joins

- A while back we introduced a new feature
  - Recommend restaurants based on the user's friends (and friends of friends)
  - It's killing us with joins

- What about sharding?
- What about SSDs?

Polyglot persistence (Not Only SQL)

- Data sets are increasingly less uniform
- Parts of Mike's data fits well in an RDBMS
- But parts of it is graph-shaped
  - If fits much better in a graph database like Neo4j!
- But what does the code look like?

An intervention!

`There shall be code.`

Conclusion

- There's an explosion of 'nosql' databases out there
  - Some are immature and experimental
  - Some are coming out of years of battle-hardened production
- NOSQL is about finding the right tool for the job
  - Frequently that's an RDBMS
  - But increasingly commonly an RDBMS is the perfect fit
- We **will** have heterogenous data backends in the future
  - Now the rest of the stack needs to step up and help developers cope with that

Key takeaway

Not Only SQL

http://neotechnology.com